RELATÓRIO TÉCNICO - RECURSOS DE NUVEM

Recomendação da Controladoria Geral da União (CGU):

Disponibilizar template de documento ou ferramenta (ex.: planilha contendo esqueleto de itens ou colunas a serem consideradas) para apoiar os órgãos contratantes no mapeamento de dependências dos serviços hospedados em nuvem, de componentes nativos do provedor e mapeamento de softwares de terceiros utilizados a partir do marketplace do provedor.

1. RECOMENDAÇÕES TÉCNICAS PARA ADEQUAÇÃO AOS REQUISITOS DE AUDITORIA

Esta seção apresenta recomendações técnicas voltadas à adoção e gestão de recursos em ambientes de computação em nuvem, com foco na conformidade com requisitos de auditoria. São abordadas práticas que promovem o uso de soluções baseadas em tecnologias abertas e plataformas não proprietárias, visando:

- Reduzir riscos de dependência tecnológica (vendor lock-in);
- Facilitar a portabilidade de aplicações e dados entre diferentes provedores de nuvem ou ambientes locais (on-premises);
- Apoiar processos futuros de migração com maior autonomia e controle por parte dos órgãos contratantes;

As orientações aqui propostas também contemplam a identificação e mapeamento de componentes nativos do provedor, bem como o uso de softwares de terceiros disponibilizados via marketplace, com o objetivo de ampliar a transparência e rastreabilidade dos serviços utilizados.

Importante destacar que as recomendações descritas têm caráter orientativo e não substituem diretrizes institucionais ou normativas específicas de cada órgão. Sua implementação exige o envolvimento de um corpo técnico qualificado, com conhecimento em arquitetura de nuvem, segurança da informação, integração de sistemas e gestão de contratos tecnológicos. Recomenda-se que sua adoção ocorra de forma planejada, colaborativa e alinhada aos objetivos estratégicos da organização pública contratante.

Recurso	Recomendação/Boa prática	Referências
Infraestrutura e Processos	A implantação de um processo DevOps visa à automatização e segurança da entrada em produção de aplicações, estabelecendo um pipeline completo de integração e entrega contínua (CI/CD). Recomenda-se a utilização do Terraform como base para infraestrutura como código (IaC), permitindo a portabilidade entre diferentes provedores de nuvem e evitando o aprisionamento tecnológico (vendor lock-in). A abordagem deve incluir: • **Automação de infraestrutura**: uso de ferramentas como o Terraform, com definição declarativa de recursos, promovendo versionamento e independência do provedor. • **Conteinerização de aplicações**: utilização do Docker para padronizar ambientes de desenvolvimento, teste e produção, facilitando a portabilidade entre provedores. • **Orquestração independente**: implementação do Kubernetes para gerenciamento de contêineres de forma agnóstica ao provedor de nuvem. • **Pipelines de CI/CD agnósticos**: configuração de pipelines utilizando ferramentas como Jenkins, GitLab CI ou GitHub Actions, que possam ser executados em qualquer ambiente. • **Gestão de configuração**: utilização de ferramentas de código aberto para gerenciamento de configurações, que não sejam específicas de um provedor. • **Documentação do processo**: manutenção de documentação detalhada sobre todo o pipeline DevOps, incluindo instruções para migração entre provedores, se necessário.	https://www.terraform.io/ https://kubernetes.io/pt-br/ https://www.jenkins.io/ https://gitlab.com/

Recurso	Recomendação/Boa prática	Referências
Infraestrutura (CD)	Serviços como Content Delivery Network (CDN) devem ser devidamente mapeados, com ênfase na documentação detalhada das regras e configurações aplicadas. Considerando que os serviços de CDN são altamente específicos a cada provedor, é fundamental garantir que esse mapeamento permita uma eventual transição controlada e segura para outro ambiente de nuvem. O inventário técnico deve conter a descrição completa de todas as regras e parâmetros configurados no serviço de CDN, incluindo, mas **não se limitando a**: • Políticas de cache e tempo de expiração de conteúdo (TTL). • Configurações de HTTPS, incluindo a gestão de certificados SSL. • Regras de redirecionamento e reescrita de URLs. • Políticas de geolocalização e restrições de acesso por região. • Configurações associadas ao WAF (Web Application Firewall), quando aplicável. Sugere-se a utilização de serviços com documentação pública e suporte a práticas abertas, como: • CDNs amplamente utilizadas (com documentação robusta e exportação de regras). • Políticas de geolocalização e restrições de acesso por região. • Soluções gratuitas de código aberto e comunitário, como **jsDelivr**, em casos aplicáveis.	https://aws.amazon.com/pt/cloudfront/ https://azure.microsoft.com/pt-br/products/cdn https://cloud.google.com/cdn https://www.cloudflare.com/ https://www.jsdelivr.com/
Infraestrutura e Sistemas	Na construção de máquinas virtuais, deve-se priorizar, sempre que possível, o uso de sistemas operacionais de código aberto, como CentOS, Debian e Ubuntu. Essa abordagem favorece a independência tecnológica, reduz custos e amplia a portabilidade entre ambientes de infraestrutura. Nos casos em que o uso de software livre não for tecnicamente viável — por exigências específicas das aplicações ou limitações de compatibilidade — recomenda-se a utilização de licenças de sistemas operacionais proprietários que atendam aos seguintes critérios: • **Portabilidade entre provedores de nuvem**: a licença deve permitir a utilização do sistema operacional em diferentes ambientes, evitando vínculo com um único fornecedor; • **Modelo de mobilidade de licenciamento (BYOL – Bring Your Own License)** : a organização deve ter autonomia para aplicar sua licença independentemente da plataforma utilizada; • **Ausência de dependência exclusiva**: a licença não deve estar atrelada exclusivamente a um provedor específico; • **Políticas claras de transição**: a documentação da licença deve prever regras explícitas para movimentação entre ambientes de nuvem ou para retorno a infraestruturas locais (on-premises). O uso de **imagens customizadas e repositórios internos de imagens homologadas** também é recomendado para padronização, segurança e agilidade no provisionamento.	https://www.centos.org https://www.debian.org/index.pt.html https://ubuntu.com/

Recurso	Recomendação/Boa prática	Referências
Banco de Dados	A adoção de Sistemas Gerenciadores de Banco de Dados (SGBDs) de código aberto deve ser priorizada como estratégia central para garantir portabilidade, controle técnico e independência de fornecedores. Essa abordagem reduz custos com licenciamento, amplia a capacidade de personalização e mitiga riscos de aprisionamento tecnológico (*vendor lock-in*). **Boas práticas recomendadas:** • **Prevenção de Dependências Proprietárias:** não utilizar extensões, operadores ou APIs exclusivas de soluções fechadas; ◦ Evitar o uso de SGBDs gerenciados com características exclusivas dos provedores de nuvem. ◦ Não utilizar extensões ou funcionalidades proprietárias que não possuam equivalentes em soluções open source. ◦ Optar por soluções amplamente compatíveis como PostgreSQL, MariaDB ou MySQL em versões comunitárias. • **Camadas de Abstração e Portabilidade** ◦ Utilizar interfaces padronizadas de acesso a dados, como JDBC/ODBC, para garantir compatibilidade entre diferentes plataformas. ◦ Desenvolver scripts de migração, carga e transformação de dados que sejam independentes do provedor ou serviço específico. ◦ Manter a documentação atualizada dos esquemas, relacionamentos e modelos de dados utilizados, facilitando eventuais migrações. • **Automação e Infraestrutura como Código (IaC)** ◦ Gerenciar a configuração e o provisionamento dos bancos de dados via ferramentas de IaC, como Terraform, garantindo reprodutibilidade e rastreabilidade. ◦ Implementar rotinas automatizadas para: ▪ Provisionamento de instâncias de banco; ▪ Backups periódicos e versionados; ▪ Recuperação em caso de falhas (Disaster Recovery). • **Estratégias de Migração e Contingência** ◦ Realizar testes regulares de portabilidade de dados entre diferentes SGBDs para validar a independência do ambiente. ◦ Manter scripts de conversão de esquemas e dados atualizados e testados. ◦ Documentar de forma detalhada os procedimentos de migração completa entre provedores, incluindo cenários de rollback e contingência.	https://www.postgresql.org/ https://www.mysql.com/ https://mariadb.com/ https://www.mongodb.com/pt-br
Infraestrutura **(Orquestração de Containers)**	A orquestração de contêineres deve ser implementada com base em plataformas agnósticas de provedor e conformes ao Kubernetes padrão (upstream), garantindo portabilidade, autonomia técnica e neutralidade tecnológica. Evita-se, assim, o uso de serviços gerenciados que introduzem dependências e dificultam a migração entre ambientes de nuvem ou infraestrutura local (*on-premises*). **Boas práticas recomendadas** • Utilizar distribuições autogerenciadas do Kubernetes, como: ◦ Kubernetes upstream; ◦ K3s (versão leve e otimizada); ◦ Rancher Kubernetes; ◦ OKD (versão open source do OpenShift); ◦ Observação: Avaliar o nível de complexidade e as necessidades do ambiente para selecionar a distribuição mais adequada • Portabilidade e Conformidade com o Kubernetes Upstream ◦ Evitar extensões proprietárias, operadores específicos ou APIs que não façam parte do ecossistema aberto do Kubernetes; ◦ Utilizar apenas controladores, *Custom Resource Definitions (CRDs)*, plugins de rede (CNI), volumes persistentes (PV/PVC) e ingress controllers amplamente suportados e independentes de fornecedor; • Provisionamento com Infraestrutura como Código (IaC): ◦ Configurar e manter a infraestrutura do cluster com ferramentas como: ▪ Terraform; ▪ Helm; ▪ Ansible.	https://kubernetes.io/pt-br/ https://k3s.io/ https://www.rancher.com/ https://okd.io/

Recurso	Recomendação/Boa prática	Referências
Infraestrutura como Código (IaC)	A utilização de Infraestrutura como Código (IaC) deve ser adotada como prática fundamental na gestão de recursos de TI, com ênfase em ferramentas não proprietárias e portáveis, como o Terraform (mantido pela HashiCorp), Pulumi ou Ansible. O uso de ferramentas como o Terraform em sua versão *open source* (sem dependências de serviços comerciais como o Terraform Cloud) deve ser priorizado, pois permite: • Provisionamento consistente e reprodutível de infraestrutura em diferentes provedores de nuvem ou ambientes on-premises. • Abstração das especificidades dos provedores, facilitando a migração e o controle das dependências. • Auditoria e versionamento completo da infraestrutura, por meio da integração com sistemas de controle de versão como Git. • Automação de ambientes complexos, com suporte a reutilização de módulos, variáveis e estruturas reutilizáveis. **Recomendações específicas** • Utilizar apenas provedores e módulos públicos padronizados, evitando extensões que dependam exclusivamente de um único fornecedor. • Estruturar os projetos com blocos reutilizáveis e parametrizáveis, para facilitar o reaproveitamento em múltiplos ambientes. • Manter a infraestrutura documentada e versionada, garantindo rastreabilidade de mudanças e compliance com requisitos de auditoria. • Integrar os scripts de IaC com pipelines de CI/CD, promovendo governança e controle nas mudanças de ambiente.	https://www.terraform.io/ https://www.pulumi.com/ https://docs.ansible.com/
APIs	No desenvolvimento, publicação e consumo de APIs em ambientes de nuvem ou híbridos, recomenda-se adotar padrões abertos e amplamente suportados, garantindo interoperabilidade, longevidade das integrações e neutralidade tecnológica. A priorização de tecnologias padronizadas evita o acoplamento a fornecedores específicos e promove maior controle técnico e auditabilidade. **Padrões Recomendados para APIs** • **Protocolos e Arquiteturas Abertas** ◦ Priorizar o uso de APIs baseadas em **RESTful** ou **GraphQL**, por serem suportadas amplamente em diferentes linguagens, frameworks e plataformas; ◦ Evitar arquiteturas proprietárias ou com extensões exclusivas não padronizadas; • **Modelagem e Documentação Padronizada** ◦ Utilizar **OpenAPI Specification (Swagger)** para definir, versionar e documentar APIs REST, facilitando a integração entre diferentes sistemas e equipes; ◦ Estruturar os contratos de API com base em modelos consistentes, versionáveis e com validação automática de parâmetros e respostas; • **Autenticação e Autorização com Padrões Abertos** ◦ Adotar mecanismos como **OAuth 2.0**, **JWT (JSON Web Token)** e **OpenID Connect**, que possuem ampla compatibilidade com bibliotecas independentes e promovem maior segurança e interoperabilidade; ◦ Evitar formatos de autenticação específicos ou acoplados a ecossistemas de fornecedores; • **Portabilidade e Independência de Plataforma** ◦ Projetar APIs para funcionamento em múltiplos ambientes (multi-cloud, on-premises, híbrido), sem exigir componentes exclusivos de infraestrutura; ◦ Realizar testes de integração e validação de portabilidade entre diferentes plataformas e stacks tecnológicos; • **Governança e Observabilidade** ◦ Estabelecer práticas de versionamento (v1, v2, etc.), com política clara de suporte e descontinuação de versões antigas; ◦ Monitorar uso, latência, erros e autenticações das APIs com ferramentas abertas de observabilidade; ◦ Registrar e armazenar logs de chamadas, acessos e eventos relevantes para auditoria e diagnóstico; • **Documentação e Acordos de Interface** ◦ Disponibilizar exemplos de requisições e respostas, mensagens de erro, políticas de uso e limites técnicos nas especificações públicas ou internas; ◦ Manter os contratos de API e documentação técnica versionados em repositórios acessíveis às equipes responsáveis;	https://graphql.org/ https://swagger.io/ https://oauth.net/2/ https://jwt.io/ https://token.dev/ https://openid.net/

Recurso	Recomendação/Boa prática	Referências
Armazenamento	Na escolha de soluções de armazenamento para ambientes em nuvem, híbridos ou on-premises, recomenda-se a adoção de tecnologias open source, portáveis e compatíveis com padrões amplamente adotados. Essas soluções oferecem maior controle técnico, redução de custos de licenciamento, e mitigam riscos de aprisionamento tecnológico (*vendor lock-in*). **Soluções recomendadas incluem:** • **MinIO**: compatível com a API S3, voltado ao armazenamento de objetos, altamente escalável, com suporte a replicação, criptografia e multitenancy. • **Ceph**: solução distribuída que oferece armazenamento de objetos, blocos e arquivos em um mesmo sistema, com forte resiliência e tolerância a falhas. • **SeaweedFS**: sistema leve e rápido para armazenamento de objetos em larga escala, com suporte a hierarquias de arquivos, replicação e compactação eficiente. **Critérios Técnicos para Avaliação de Soluções** • **Desempenho:** avaliar capacidade de throughput, latência e escalabilidade horizontal. • **Segurança:** suporte a criptografia de dados em trânsito e em repouso, autenticação granular, ACLs e compatibilidade com mecanismos de identidade externos. • **Operação:** facilidade de implantação, configuração, atualização e monitoramento com ferramentas padronizadas. • **Portabilidade:** compatibilidade com APIs abertas (ex: S3 REST), suporte a replicação e sincronização entre nuvens e ambientes locais. • **Conformidade e Governança:** aderência a políticas de proteção de dados, versionamento, retenção e auditoria. **Boas práticas adicionais** • Avaliar continuamente o desempenho e o uso da solução com métricas observáveis; • Integrar o armazenamento com pipelines de CI/CD e automações via infraestrutura como código (IaC); • Documentar os requisitos de armazenamento por sistema/serviço, incluindo volume, latência e política de backup; • Registrar formalmente a escolha da solução, com comparativos técnicos, critérios de decisão e impactos operacionais;	https://min.io/ https://ceph.io/en/ https://github.com/seaweedfs/seaweedfs
Mensageria	Em arquiteturas de sistemas distribuídos e microsserviços, a utilização de soluções de mensageria abertas, portáveis e independentes de fornecedor deve ser priorizada para garantir interoperabilidade, resiliência e flexibilidade em ambientes multi-nuvem e híbridos. Essas soluções devem permitir comunicação assíncrona, desacoplamento de serviços e garantia de entrega, com suporte a padrões abertos de protocolo e compatibilidade com múltiplas linguagens e plataformas. **Soluções amplamente recomendadas** • **Apache Kafka** – indicado para aplicações que exigem alta taxa de transferência, persistência de eventos e arquitetura orientada a fluxo de dados em tempo real. • **RabbitMQ** – recomendado para cenários com requisitos variados de roteamento, filas complexas, confirmação de entrega e múltiplos protocolos (AMQP, MQTT). • **NATS** – alternativa leve e eficiente para comunicação em tempo real com baixa latência e footprint reduzido, ideal para ambientes distribuídos e IoT; **Recomendações Técnicas** • **Protocolos Abertos**: utilizar **AMQP**, **MQTT**, **Kafka nativo** ou **STAN (Streaming NATS)** para garantir compatibilidade entre ambientes e evitar acoplamento a APIs restritas. • **Infraestrutura como Código (IaC)**: provisionar e configurar brokers de mensageria com ferramentas como **Terraform**, **Helm** ou **Ansible**, garantindo reprodutibilidade e controle de versão; • **Gestão e Versionamento da Configuração:** ◦ Manter documentação e versionamento das topologias de mensagens (filas, tópicos, bindings, exchanges), regras de roteamento, persistência e políticas de retenção; ◦ Automatizar testes de configuração e validação de integridade. • **Portabilidade e Testes de Migração**: validar periodicamente a capacidade de migração entre ambientes (cloud, híbrido, local), incluindo simulações de failover e análise de compatibilidade de clientes e bibliotecas. • **Monitoramento e Observabilidade**: integrar os sistemas de mensageria com ferramentas de observabilidade abertas, expondo métricas, logs e eventos relevantes à operação;	https://kafka.apache.org/ https://www.rabbitmq.com/ https://nats.io/ https://www.amqp.org/

Recurso	Recomendação/Boa prática	Referências

Monitoramento	A implementação de soluções de monitoramento, observabilidade e logging deve priorizar o uso de ferramentas open source, portáveis e compatíveis com padrões abertos, evitando o acoplamento a plataformas específicas de provedores de nuvem. Essa abordagem promove transparência operacional, auditoria contínua, portabilidade entre ambientes e redução de dependências tecnológicas.
	Ferramentas Recomendadas
	• **Prometheus** – coleta e armazenamento de métricas em tempo real com modelo de dados multidimensional e linguagem de consulta (PromQL);
	• **Grafana** – visualização de métricas e logs com dashboards personalizáveis, alertas e integração com múltiplas fontes de dados;
	• **OpenTelemetry** – framework unificado para instrumentação, coleta e exportação de métricas, logs e *traces*, com suporte a diversos protocolos e formatos.
	Boas Práticas Recomendadas
	• **Centralização e padronização da observabilidade**: coletar métricas, logs e *traces* em uma arquitetura unificada, utilizando protocolos abertos como OTLP e formatos interoperáveis (JSON, YAML, Protobuf);
	• **Automação e reprodutibilidade**: gerenciar a configuração das ferramentas por meio de infraestrutura como código (IaC), garantindo versionamento, replicação e auditoria das definições de monitoramento;
	• **Compatibilidade entre ambientes**: utilizar exportadores, agentes e coletores compatíveis com ambientes em nuvem pública, privada e *on-premises*.
	• **Governança e rastreabilidade**:
	  ◦ Versionar dashboards, regras de alerta e definições de *health checks*, armazenando-os em repositórios controlados por Git;
	  ◦ Documentar a estratégia de observabilidade, incluindo indicadores-chave de desempenho (KPIs), requisitos de conformidade e métricas críticas de segurança, desempenho e custo.
	• **Testes e validação contínua**: validar a efetividade dos alertas, a cobertura de monitoramento e a integridade dos dados coletados por meio de simulações, testes automatizados e auditorias periódicas..
		https://prometheus.io/
		https://grafana.com/
		https://opentelemetry.io/

Backup	A adoção de uma estratégia de backup e recuperação de desastres (Disaster Recovery) baseada em soluções não proprietárias, portáveis e compatíveis com múltiplos ambientes é essencial para garantir a continuidade dos serviços e minimizar riscos operacionais, regulatórios e de perda de dados.
	Essa estratégia deve prever criptografia, versionamento, testes regulares de restauração e documentação completa, promovendo resiliência e controle.
	Ferramentas recomendadas
	• **Velero** – voltado para backup e restauração de clusters Kubernetes, com suporte a volumes persistentes e objetos do cluster.
	• **Rclone** – ideal para sincronização, cópia, espelhamento e criptografia de arquivos entre diferentes serviços de armazenamento e ambientes locais.
	• **Restic** – ferramenta de backup rápida, segura e eficiente, com deduplicação, compressão e criptografia ponta a ponta.
	• **Duplicacy** – oferece backups incrementais, criptografados e com suporte a múltiplos destinos simultâneos (cloud, local, remoto), com gerenciamento de snapshots.
	• **BorgBackup** – solução robusta para backup criptografado e deduplicado, indicada para servidores e ambientes crítico.
	Recomendações Estratégicas
	• **Redundância entre Ambientes**: implementar replicação automática entre diferentes zonas, provedores ou entre nuvem e on-premises, sempre que tecnicamente viável.
	• **Segurança e Confiabilidade**: garantir que os backups sejam criptografados, versionados e testados periodicamente, com simulações documentadas de recuperação.
	• **Automação via Infraestrutura como Código (IaC)**: definir políticas de backup, agendamentos, retenção e pontos de recuperação com ferramentas como Terraform, Ansible ou Helm;
	• Manter um repositório seguro contendo:
	  ◦ Documentação das estratégias de backup e recuperação.
	  ◦ Scripts de automação.
	  ◦ SOPs (Procedimentos Operacionais Padrão) para restauração em diferentes cenários.
	• Avaliar e documentar os requisitos de RTO (Recovery Time Objective) e RPO (Recovery Point Objective) para cada sistema ou serviço crítico.
	• Monitorar o desempenho das execuções de backup com métricas de sucesso, tempo, integridade e espaço utilizado.
		https://velero.io/
		https://rclone.org/
		https://restic.net/
		https://duplicacy.com/
		https://www.borgbackup.org/

Recurso	Recomendação/Boa prática	Referências
Segurança	A arquitetura de segurança em ambientes de nuvem deve priorizar o uso de soluções que garantam independência de provedor, gestão centralizada e auditável de identidades, e definição de políticas de acesso declarativas, promovendo transparência, interoperabilidade e conformidade regulatória. **Soluções abertas e multiambiente recomendadas** • **Keycloak** – plataforma open source para gestão de identidade e acesso (IAM), com suporte a SSO, OAuth2, OpenID Connect e integração com diretórios corporativos (LDAP, AD). • **Auth0** e **Okta** – plataformas SaaS amplamente compatíveis com múltiplos ambientes e provedores, que oferecem recursos avançados de autenticação sem dependência de ecossistemas fechados. • **HashiCorp Vault** – ferramenta robusta para gestão segura e granular de segredos, tokens, certificados e credenciais, com criptografia em repouso e em trânsito, além de controle de auditoria. • **Open Policy Agent (OPA)** – motor de políticas open source que permite definir, versionar e validar regras de segurança como código (*Policy as Code*), integrado a aplicações, pipelines, gateways, containers e infraestrutura. **Boas práticas técnicas recomendadas** • **Centralização da gestão de segredos e credenciais**: utilizar o HashiCorp Vault para gerenciamento seguro com rotação automática, criptografia e segregação por ambiente. • **Políticas como Código (PaC)**: definir e versionar políticas de acesso e autorização usando OPA com a linguagem Rego, promovendo automação, validação e testes contínuos. • **Integração multiambiente e governança unificada**: garantir que a autenticação e a autorização sejam aplicáveis em ambientes públicos, privados e híbridos, com camadas de política reutilizáveis. • **Auditoria e conformidade regulatória**: Armazenar eventos de autenticação, logs de acesso e alterações em políticas em repositórios protegidos, com retenção conforme exigido por normas como LGPD, ISO 27001 e PCI-DSS. • **Segurança na esteira de desenvolvimento (DevSecOps)**: Integrar IAM, Vault e OPA a pipelines de CI/CD, promovendo segurança desde a fase de codificação até o deploy e operação.	https://www.keycloak.org/ https://auth0.com/ https://www.okta.com/ https://www.hashicorp.com/pt/products/vault https://www.openpolicyagent.org/
Automação de Pipelines	A automação de pipelines de Integração Contínua (CI) e Entrega Contínua (CD) deve ser realizada com ferramentas abertas, portáveis e amplamente suportadas, compatíveis com múltiplos ambientes (nuvem, on-premises, híbrido). Essa estratégia garante rastreabilidade, controle de versões, segurança e governança de mudanças, reduzindo a dependência de serviços específicos de fornecedores. **Soluções recomendadas:** • **Jenkins** – ferramenta extensível, modular e com ecossistema maduro de plugins, amplamente usada para CI/CD. • **GitHub Actions** – nativa da plataforma GitHub, com suporte a automações reutilizáveis, eventos personalizados e execução local ou distribuída. • **GitLab CI/CD** – pipeline nativo da plataforma GitLab, com integração total ao versionamento de código e suporte a *runners* autogerenciados. • **ArgoCD** – ferramenta declarativa para CD com GitOps em ambientes Kubernetes, ideal para clusters multi-cloud com foco em segurança e controle de estado. **Boas Práticas Recomendadas** • **Versionamento e rastreabilidade**: declarar pipelines em arquivos YAML versionados com o código-fonte (repositórios Git), garantindo governança e histórico de alterações. • **Execução em ambientes controlados**: utilizar *runners* ou *executors* autogerenciados e portáveis, permitindo flexibilidade, isolamento e menor dependência de ambientes gerenciadosa. • **Validações e automações integradas**: integrar ferramentas de análise estática de código, escaneamento de segurança, linting e testes automatizados à esteira de CI/CD. • **Revisões e políticas de mudança**: submeter alterações de pipelines a revisão por pares (pull/merge requests), aplicando controles de aprovação e verificação automatizada. • **Governança e auditoria contínuas**: documentar e versionar todas as fases do pipeline, incluindo build, teste, empacotamento, deploy e rollback; implementar alertas, métricas de execução e geração de evidências para auditorias técnicas e regulatórias. • **Padronização e reutilização**: criar *templates*, *workflows* ou *módulos* reutilizáveis para padronizar fluxos entre equipes e serviços.	https://www.jenkins.io/ https://github.com/features/actions https://docs.gitlab.com/ https://argoproj.github.io/cd/

Recurso	Recomendação/Boa prática	Referências
Gereciamento de Configuração	O gerenciamento de configurações em ambientes distribuídos deve ser realizado com ferramentas open source, autogerenciáveis e independentes de provedor, garantindo alta disponibilidade, consistência e portabilidade em múltiplos ambientes (nuvem, on-premises e híbridos). Essas ferramentas devem possibilitar a atualização dinâmica, auditoria de alterações e integração com esteiras de CI/CD, promovendo governança e automação na gestão de parâmetros, flags e ajustes operacionais. **Soluções recomendadas** • **Consul** – ferramenta da HashiCorp para descoberta de serviços, armazenamento de configurações chave-valor, integração com *service mesh*, ACLs e DNS interno. • **etcd** – armazenamento chave-valor distribuído com alta consistência, utilizado como backend em soluções como Kubernetes. • **Apache Zookeeper** – sistema de coordenação e armazenamento de configuração para aplicações distribuídas, com foco em consistência, resiliência e replicação automática. **Boas Práticas Recomendadas** • **Separação entre dados sensíveis e operacionais**: armazenar configurações sensíveis (segredos, tokens, certificados) separadamente, integrando com soluções como HashiCorp Vault. • **Atualização dinâmica e notificações em tempo real**: priorizar ferramentas que suportem atualização de configurações sem reinício da aplicação (*hot reload*), com notificações para serviços dependentes. • **Integração com CI/CD**: definir o gerenciamento de configuração como parte integrante do ciclo de deploy automatizado, com versionamento, validação sintática e rollback em caso de erro. • **Alta disponibilidade e replicação**: garantir replicação síncrona ou assíncrona entre zonas, clusters ou regiões, assegurando tolerância a falhas e consistência. • **Auditoria e rastreabilidade**: registrar alterações de chave/valor, logar quem alterou, quando e com qual finalidade. Integrar com ferramentas de observabilidade e geração de alertas. • **Governança e documentação**: mapear todos os serviços dependentes de configurações dinâmicas, versionar os arquivos e manter repositórios Git específicos para configuração.	https://developer.hashicorp.com/consul https://etcd.io/ https://zookeeper.apache.org/
Análise e Processamento de Dados	Para cargas de trabalho envolvendo análise e processamento de grandes volumes de dados (big data), devem ser priorizadas soluções open source, escaláveis e compatíveis com múltiplos ambientes, em detrimento de serviços analíticos proprietários fortemente acoplados aos provedores de nuvem. **Tecnologias Recomendadas** • **Apache Spark** – framework distribuído para processamento em lote e em tempo real, com suporte a diversos formatos (Parquet, ORC, JSON), linguagens (Python, Scala, SQL) e integrações (Kafka, Hadoop, Hive, JDBC); • **Presto** / **Trino** – motores SQL distribuídos para análise de dados em múltiplas fontes (data lakes, bancos relacionais, NoSQL), com baixa latência e suporte a conectores plugáveis. **Boas Práticas Recomendadas** • **Centralização em formatos abertos**: Armazenar dados em **data lakes** estruturados sobre formatos abertos e eficientes como **Parquet**, **ORC** ou **Avro** para maximizar compatibilidade e compressão. • **Execução distribuída e tolerância a falhas**: Utilizar ferramentas que suportem **execução paralela**, **reprocessamento automático** e **escalonamento horizontal** dos *workers.* • **Orquestração de pipelines de dados**: Empregar Apache Airflow, Dagster ou Prefect para gerenciamento de DAGs, dependências, agendamento e monitoramento de tarefas de ETL/ELT; Garantir rastreabilidade completa de cada etapa dos fluxos de dados. • **Versionamento e reprodutibilidade**: Versionar os scripts de transformação, schemas, metadados e modelos de machine learning atrelados ao pipeline. • **Infraestrutura como Código e elasticidade**: Automatizar o provisionamento dos clusters de processamento com ferramentas como Terraform, Helm ou Ansible; Implementar escalonamento automático com base em métricas de utilização (CPU, memória, jobs pendentes). • **Auditoria e observabilidade**: Integrar o ambiente com Prometheus, Grafana e ferramentas de logs para garantir métricas de ingestão, throughput e latência. • **Segurança e compliance**: Garantir que os dados estejam criptografados, mascarados quando necessário, e que haja controle de acesso auditável nas fontes e nos resultados.	https://spark.apache.org/ https://prestodb.io/ https://trino.io/

Recurso	Recomendação/Boa prática	Referências
Armazenamento de Data Lakes	A construção e a gestão de data lakes modernos devem se basear em tecnologias open source, com foco em governança de metadados, versionamento de dados, interoperabilidade e portabilidade entre motores e ambientes. A arquitetura deve possibilitar consultas distribuídas, transações ACID, e integração com ferramentas analíticas e de machine learning. **Tecnologias Recomendadas** • **Apache Iceberg** – projeto open source que oferece gerenciamento de tabelas para data lakes com suporte a versionamento, transações ACID, schema evolution, otimizações de leitura/escrita e compatibilidade com Presto, Trino, Spark e Flink; • **Delta Lake** – camada de armazenamento criada pela Databricks, com suporte a transações ACID, controle de esquema, e integração com Apache Spark para processamento escalável e confiável; • **Apache Hudi** – framework para ingestão incremental, gerenciamento de versões e controle de mutações em data lakes, com suporte a compactação de dados, rollback e integração com motores distribuídos como Hive, Presto e Trino. **Boas Práticas Recomendadas** • **Adoção de formatos abertos e comprimidos**: utilizar formatos como **Parquet** e **ORC** para garantir alta performance, compressão eficiente e compatibilidade entre motores de leitura; • **Governança e rastreabilidade**: implementar camadas de controle de acesso, versionamento de dados e *lineage* com ferramentas que suportem logs de alterações, auditoria e reprocessamento histórico; • **DataOps e automação**: integrar pipelines de ingestão, transformação e compactação de dados com ferramentas de orquestração como Apache Airflow, Prefect ou Dagster, tratando a estrutura do lake como código; • **Portabilidade e consistência entre motores**: validar periodicamente a consistência dos dados ao migrar entre Trino, Spark, Flink e Hive, testando queries equivalentes, schemas e versões de tabelas; • **Documentação e versionamento de metadados**: versionar o catálogo de tabelas, esquemas, partições e camadas lógicas do lake, integrando com repositórios Git e ferramentas de metastore (Glue-compatible, Hive, etc.). • **Escalabilidade e separação de camadas**: organizar a arquitetura em camadas lógicas (*raw*, *refined*, *consumption*) e físicas (storage + metastore), com separação clara entre ingestão, transformação e acesso analítico.	https://iceberg.apache.org/ https://delta.io/ https://hudi.apache.org/
Machine Learning	Na construção, treinamento, validação e operação de modelos de Machine Learning (ML), é fundamental priorizar plataformas abertas, modulares e portáveis, que garantam controle sobre os ciclos de vida dos modelos, interoperabilidade entre ambientes e independência de provedor. **Plataformas Recomendadas** • **TensorFlow** – biblioteca open source amplamente adotada para construção e treinamento de modelos de aprendizado de máquina e deep learning, com suporte a múltiplas linguagens (Python, Java, C++) e execução distribuída. • **Kubeflow** – plataforma nativa de Kubernetes para orquestração de pipelines de ML com suporte a experiment tracking, deploy, gerenciamento de versões de modelos e automação do ciclo de vida. • **MLflow** – sistema modular que oferece rastreamento de experimentos, versionamento de modelos, repositório de artefatos, controle de ambientes e deploy multiplataforma. **Boas Práticas Recomendadas** • Manter pipelines de ML definidos como código, integrados a repositórios Git e ferramentas de CI/CD. • Utilizar ambientes de execução portáveis (Docker) com descrição explícita de dependências. • Implementar versionamento de datasets, experimentos, hiperparâmetros e modelos treinados com ferramentas como DVC ou MLflow. • Garantir que o ciclo de vida dos modelos (treinamento, validação, inferência e monitoramento) seja automatizado e auditável. • Utilizar recursos de Kubernetes e ferramentas como Argo Workflows ou Kubeflow Pipelines para orquestração distribuída e escalável.	https://www.tensorflow.org/ https://www.kubeflow.org/ https://mlflow.org/

Recurso	Recomendação/Boa prática	Referências
Governança e Otimização de Custos	Para garantir uma gestão eficiente, transparente e neutra dos custos em ambientes de nuvem, é essencial utilizar ferramentas abertas, portáveis e compatíveis com ambientes multi-cloud e on-premises. Tais ferramentas devem permitir monitoramento contínuo, análise preditiva, controle orçamentário e responsabilização por consumo. **Ferramentas Recomendadas** • **OpenCost** – solução mantida pela CNCF que permite análise de custos detalhados em ambientes Kubernetes, com granularidade por namespace, workload e aplicação. • **Kubecost (versão open source)** – fornece rastreamento de custos por recurso em Kubernetes, incluindo métricas de eficiência e desperdício. • **Infracost (open core)** – CLI gratuita para exibir estimativas de custo diretamente no código Terraform, antes da aplicação de mudanças. • **Grafana + Prometheus** – visualização de métricas e construção de dashboards customizados com dados de custos extraídos via APIs abertas; • **Apache Superset** – ferramenta analítica para consultas interativas e visualização de dados de custo extraídos de múltiplas fontes. **Boas Práticas Recomendadas** • **Unificação de dados de custo**: consolidar exportações de billing e métricas de uso em camadas analíticas internas, usando ferramentas como DuckDB, Superset ou notebooks Jupyter. • **Estimativa antecipada com IaC**: Integrar o **Infracost** a pipelines CI/CD para calcular impacto financeiro de mudanças antes do *terraform apply*. • **Dashboards gratuitos e portáveis**: utilizar **Grafana** com **Prometheus** para montar painéis que exibem consumo de CPU, memória, egress, storage e custo estimado por aplicação ou cluster • **Auditorias periódicas e alertas gratuitos**: Configurar alertas em Prometheus para identificar desvios de comportamento ou picos de custo atípicos; Registrar ociosidade de recursos com base em métricas de uso real; • **Políticas e cotas operacionais:** definir cotas informativas por namespace, projeto ou equipe, com relatórios exportáveis e rastreáveis em Git ou S3. • **Integração com ferramentas já utilizadas**: reutilizar infraestrutura existente (Kubernetes, Terraform, Git) para acoplar controle de custo à esteira técnica sem ferramentas pagas.	https://www.opencost.io/ https://www.kubecost.com/ https://www.infracost.io/ https://grafana.com/ https://prometheus.io/ https://superset.apache.org/
Rastreamento de Enventos de Segurança	A implementação de mecanismos de rastreamento e auditoria de eventos de segurança deve garantir monitoramento contínuo, detecção proativa de ameaças e correlação de eventos em ambientes distribuídos. A arquitetura deve permitir personalização de regras, integração com fluxos de provisionamento e resposta automatizada a incidentes. **Ferramentas Recomendadas** • **Falco** – detecção de comportamento anômalo em containers e hosts Linux com base em regras de sistem; • **Wazuh** – monitoramento de integridade de arquivos, análise de logs, correlação de eventos e geração de alertas com foco em conformidade; • **OSSEC** – detecção de intrusões baseada em host com suporte a logs locais e remotos; • **TheHive** – gerenciamento de incidentes com suporte a análises colaborativas, vinculação de evidências e orquestração de resposta; **Boas Práticas Recomendadas** • **Integração com pipelines e infraestrutura como código**: conectar sensores de auditoria e segurança aos fluxos de CI/CD e provisionamento de infraestrutura para cobertura desde o deploy inicial. • **Modelagem de regras personalizadas**: criar e manter regras de detecção baseadas nos perfis de risco, aplicações utilizadas e processos operacionais críticos. • **Coleta, retenção e trilha de auditoria**: garantir que logs e eventos sejam coletados com controle de acesso, criptografia e retenção alinhada às políticas institucionais. • **Correlação multicanal de eventos**: agrupar e correlacionar alertas provenientes de sistemas operacionais, containers, redes e aplicações distribuídas para construir um cenário completo de ameaça. • **Automação de resposta a incidentes**: utilizar motores de orquestração para execução automatizada de respostas, como bloqueio de acesso, notificação, coleta de evidências ou abertura de tickets. • **Conformidade e relatórios técnicos**: gerar relatórios periódicos com evidências rastreáveis, indicadores de segurança e histórico de intervenções para apoio a auditorias e certificações.	https://falco.org/ https://wazuh.com/ https://www.ossec.net/ https://thehive-project.org/

Recurso	Recomendação/Boa prática	Referências
Armazenamento e Versionamento de Código-Fonte	O gerenciamento de código-fonte deve ser estruturado sobre plataformas amplamente compatíveis e integráveis com fluxos de CI/CD, de modo a garantir portabilidade entre ambientes, controle de acesso granular, automação contínua e rastreabilidade de alterações. **Plataformas Recomendadas** • **GitHub** – oferece repositórios Git, controle de permissões, automações via Actions e integração com sistemas externos de auditoria, segurança e deploy. • **GitLab** – solução completa com repositório Git, CI/CD integrado, controle de acesso, gerenciamento de issues e compatibilidade com Kubernetes e containers. **Boas Práticas Recomendadas** • **Uso de protocolos e formatos abertos**: adotar Git sobre SSH ou HTTPS, garantindo compatibilidade com qualquer ferramenta de DevOps ou IDE. • **Integração com CI/CD e segurança**: conectar repositórios aos pipelines de integração e entrega contínua, ferramentas de análise estática, scanners de vulnerabilidade e validação de infraestrutura como código. • **Governança de acesso e auditoria**: estabelecer políticas claras de permissões, revisão de código obrigatória e registro de alterações para cada *merge* ou *push*; Monitorar atividades com logs de acesso e integrações com SIEMs ou serviços de alerta. • **Redundância e disponibilidade**: armazenar cópias espelhadas dos repositórios em múltiplas regiões ou datacenters e manter backups programados com versionamento. • **Padrões de versionamento e colaboração**: documentar práticas de branching, semântica de versões (semver), convenções de *commits*, estratégias de *release* e nomenclatura de branches. • **Automação da governança de código**: utilizar workflows automatizados para verificação de estilo, testes, cobertura de código e conformidade com requisitos técnicos e legais.	https://github.com/features/actions https://about.gitlab.com/

2. ESTRATÉGIAS DE OTIMIZAÇÃO NO USO E GESTÃO RECURSOS DE NUVEM

Esta seção apresenta recomendações técnicas voltadas à melhoria do uso e da gestão de recursos em ambientes de computação em nuvem, contemplando as principais modalidades de serviço: Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS). O foco está na identificação de práticas que viabilizem a alocação eficiente de recursos, alinhando consumo à demanda real, com vistas à redução de custos, aumento de desempenho e maior previsibilidade operacional.

As estratégias abordadas incluem ações para otimizar o provisionamento, escalabilidade, automação e monitoramento de recursos, com ênfase em eficiência técnica e financeira. A adoção dessas práticas permite às organizações desenvolver uma abordagem mais sustentável e resiliente em nuvem, garantindo melhor aproveitamento da infraestrutura disponível, além de ampliar a capacidade de governança e controle sobre o ambiente computacional.

2.1 - IaaS: Recursos de Infraestrutura como Serviço

Recurso	Descrição	Orientações	Benefícios e impactos	Links de referência

Recurso	Descrição	Orientações de Análise de Utilização:	Benefícios e impactos	Links de referência
Máquinas Virtuais	**Escolha da Máquina Virtual**: Escolher o tipo e tamanho de instância mais adequado à carga de trabalho, visando performance e custo-benefício.	**Análise de Utilização:** Monitorar continuamente o consumo de CPU, memória e armazenamento. **Ajustes Regulares:** Reavaliar e ajustar instâncias conforme a variação da demanda. **Comparação de Tipos:** Testar diferentes tipos de instâncias para encontrar o mais eficiente. **Auditorias Técnicas:** Realizar levantamentos periódicos sobre uso e desempenho. **Ferramentas Automatizadas:** Implementar soluções de monitoramento e escalonamento automático. **Revisão de Políticas:** Atualizar diretrizes de uso conforme as melhores práticas e mudanças na carga. **Scripts de Automação:** Criar rotinas para desligamento automático, redimensionamento e execução de tarefas recorrentes. **Auto-Scaling Dinâmico:** Utilizar recursos que ajustem a capacidade com base em métricas de uso em tempo real. **Monitoramento Inteligente:** Analisar logs para identificar padrões de uso e definir horários ideais para operações. **Ajustes Baseados em Dados:** Refinar a alocação de recursos com base em análises periódicas. **Agendamentos e Alertas:** Integrar mecanismos que disparem alertas e executem ações automaticamente conforme condições pré-definidas. **Revisão de Efetividade:** Avaliar regularmente a eficácia das automações implementadas	Otimização de custos operacionais Aumento da eficiência de recursos Desempenho aprimorado Redução de desperdício Flexibilidade e escalabilidade	AWS Instance Types https://aws.amazon.com/pt/ec2/instance-types/ Máquinas virtuais no Azure https://learn.microsoft.com/pt-br/azure/virtual-machines/ Google Compute Engine Machine Types https://cloud.google.com/compute/docs/machine-types?hl=pt-br
Máquinas Virtuais **Máquinas Virtuais**	**Escolha da Máquina Virtual**: Escolher o tipo e tamanho de instância mais adequado à carga de trabalho, visando performance e custo-benefício. **Automação das Máquinas Virtuais:** Automatizar o desligamento e redimensionamento de VMs em períodos de baixa demanda.	**Análise de Utilização:** Monitorar continuamente o consumo de CPU, memória e armazenamento. **Ajustes Regulares:** Reavaliar e ajustar instâncias conforme a variação da demanda. **Comparação de Tipos:** Testar diferentes tipos de instâncias para encontrar o mais eficiente. **Auditorias Técnicas:** Realizar levantamentos periódicos sobre uso e desempenho. **Ferramentas Automatizadas:** Implementar soluções de monitoramento e escalonamento automático. **Revisão de Políticas:** Atualizar diretrizes de uso conforme as melhores práticas e mudanças na carga. **Scripts de Automação:** Criar rotinas para desligamento automático, redimensionamento e execução de tarefas recorrentes. **Auto-Scaling Dinâmico:** Utilizar recursos que ajustem a capacidade com base em métricas de uso em tempo real. **Monitoramento Inteligente:** Analisar logs para identificar padrões de uso e definir horários ideais para operações. **Ajustes Baseados em Dados:** Refinar a alocação de recursos com base em análises periódicas. **Agendamentos e Alertas:** Integrar mecanismos que disparem alertas e executem ações automaticamente conforme condições pré-definidas. **Revisão de Efetividade:** Avaliar regularmente a eficácia das automações implementadas	Otimização de custos operacionais Aumento da eficiência de recursos Desempenho aprimorado Redução de desperdício Redução de custos significativa Flexibilidade e escalabilidade Melhoria na gestão de operações Conformidade com políticas de TI	AWS Instance Types https://aws.amazon.com/pt/ec2/instance-types/ Máquinas virtuais no Azure https://learn.microsoft.com/pt-br/azure/virtual-machines/ Google Compute Engine Machine Types https://cloud.google.com/compute/docs/machine-types?hl=pt-br AWS Instance Scheduler https://aws.amazon.com/pt/solutions/implementations/instance-scheduler-on-aws/ Azure Automation https://azure.microsoft.com/pt-br/products/automation/ Gerenciamento VM Google Cloud https://cloud.google.com/compute/docs/instances/schedule-instance-start-stop?hl=pt-br

Recurso	Descrição	Orientações	Benefícios e impactos	Links de referência
		Atualização Constante: Implementar atualizações automáticas para sistemas operacionais e softwares, garantindo que estejam sempre na versão mais recente. **Aplicação de Patches de Segurança**: Instalar patches regularmente para corrigir vulnerabilidades e proteger o ambiente. **Configuração Segura do Ambiente**: Estabelecer configurações que priorizem a segurança e a eficiência operacional. **Monitoramento de Desempenho**: Acompanhar o desempenho do software para identificar e resolver problemas rapidamente. **Ajustes Regulares:** Reavaliar e ajustar software para garantir conformidade e reduzir custos. **Ferramentas Automatizadas:** Utilizar ferramentas que ajudem na administração e monitoramento de forma contínua e automática. **Revisão de Políticas:** Atualizar diretrizes de uso conforme as melhores práticas e mudanças na carga.	Segurança aprimorada Maior desempenho Compatibilidade entre softwares Redução de vulnerabilidades Otimização Facilitação da gestão de sistemas Redução de custos operacionais Otimização de custos operacionais Aumento da eficiência de recursos Desempenho aprimorado Redução de desperdício Flexibilidade e escalabilidade	AWS Compute Optimizer https://aws.amazon.com/pt/compute-optimizer/ Azure - Microsoft Learn https://learn.microsoft.com/pt-br/azure/virtual-machines/cost-optimization-best-practices Melhores práticas para otimizar seus custos Google Cloud https://cloud.google.com/blog/topics/cost-management/best-practices-for-optimizing-your-cloud-costs AWS Instance Types https://aws.amazon.com/pt/ec2/instance-types/ Máquinas virtuais no Azure https://learn.microsoft.com/pt-br/azure/virtual-machines/ Google Compute Engine Machine Types https://cloud.google.com/compute/docs/machine-types?hl=pt-br
Máquinas Virtuais	**Otimização de Sistema Operacional e Software:** Manutenção do SO e software das VMs para garantir eficiência e segurança. **Escolha da Máquina Virtual**: Escolher o tipo e tamanho de instância mais adequado à carga de trabalho, visando performance e custo-benefício.	**Gerenciamento de Configurações:** Utilizar ferramentas que ajudem na administração e configuração do ambiente, melhorando a eficiência. **Auditorias Regulares de Segurança:** Realizar auditorias periódicas para identificar e corrigir riscos. **Testes de Carga:** Realizar testes de carga para avaliar a capacidade e o desempenho. **Comparação de Tipos:** Testar diferentes tipos de instâncias para encontrar o mais eficiente. **Monitoramento de Segurança Contínua:** Implementar soluções de monitoramento de segurança contínua para detectar ameaças. **Escalonamento Automático:** Configurar escalonamento automático para ajustar recursos com base na demanda.		
		Scripts de Automação: Criar rotinas para desligamento automático, redimensionamento e execução de tarefas recorrentes. **Auto-Scaling Dinâmico:** Utilizar recursos que ajustem a capacidade com base em métricas de uso em tempo real.		

Recurso	Descrição	Orientações	Benefícios e impactos	Links de referência
Máquinas Virtuais	**Compra e Uso de Instâncias Reservadas:** Aquisição de instâncias reservadas para cargas de trabalho previsíveis e de longo prazo, aproveitando descontos significativos.	**Avaliação de Padrões de Uso**: Analisar os padrões atuais de uso para identificar necessidades futuras, auxiliando na tomada de decisões sobre a compra de instâncias reservadas e/ou on-demand. Considerar o custo-benefício e realizar comparações de preços. **Revisão Anual da Estratégia**: Realizar uma revisão anual da estratégia de uso de instâncias reservadas, baseando-se nas mudanças nas necessidades de negócios para garantir alinhamento e eficiência.	Redução substancial nos custos de computação, garantia de disponibilidade de recursos, planejamento financeiro melhorado.	AWS Reserved Instances https://aws.amazon.com/pt/ec2/pricing/reserved-instances/ Azure Reserved VM Instances https://azure.microsoft.com/en-us/pricing/reserved-vm-instances/ Reservas de recursos zonais do Compute Engine Google Cloud https://cloud.google.com/compute/docs/instances/reservations-overview?hl=pt-br
Máquinas Virtuais **Máquinas Virtuais**	**Escolha da Máquina Virtual**: Escolher o tipo e tamanho de instância mais adequado à carga de trabalho, visando performance e custo-benefício. **Implementação de Multi-AZ:** Avaliação da necessidade de utilizar múltiplas zonas de disponibilidade (Multi-AZ) para garantir alta disponibilidade e resiliência.	**Análise de Viabilidade**: Avaliar a viabilidade do uso de Multi-AZ para aplicações críticas, onde a disponibilidade e a tolerância a falhas são essenciais. **Configuração de Replicação Automática**: Implementar replicação automática de dados e processos críticos entre zonas de disponibilidade para minimizar a perda de dados. **Testes Regulares:** Realizar testes periódicos de failover para assegurar que os objetivos de disponibilidade sejam atendidos. **Revisão de Política:** Revisar e ajustar a estratégia de Multi-AZ com base em mudanças na carga de trabalho e novas necessidades de negócios e tecnológicas, garantindo a adequação contínua.	Otimização de custos operacionais Aumento da eficiência de recursos Desempenho aprimorado Redução de desperdício Flexibilidade e escalabilidade	AWS Instance Types https://aws.amazon.com/pt/ec2/instance-types/ AWS Multi-AZ Deployment https://docs.aws.amazon.com/pt_br/redshift/latest/mgmt/managing-cluster-multi-az.html Máquinas virtuais no Azure https://learn.microsoft.com/pt-br/azure/virtual-machines/ Opções regionais do Azure https://learn.microsoft.com/pt-br/azure/reliability/regions-list Google Compute Engine Machine Types https://cloud.google.com/compute/docs/machine-types?hl=pt-br Locais do Google Cloud https://cloud.google.com/about/locations?hl=pt-br
		Monitoramento Contínuo: Monitorar constantemente o desempenho da implementação Multi-AZ, assegurando que os objetivos de disponibilidade sejam atendidos. **Revisão da Estratégia**: Revisar e ajustar a estratégia de Multi-AZ com base em mudanças na carga de trabalho e novas necessidades de negócios e mudanças tecnológicas, garantindo a adequação contínua. **Auditorias Técnicas:** Realizar levantamentos periódicos sobre uso e desempenho, identificando oportunidades de melhoria. **Ferramentas Automatizadas:** Implementar soluções de monitoramento e escalonamento automático. **Políticas:** Atualizar diretrizes de uso conforme as melhores práticas e mudanças tecnológicas, para garantir a adequação contínua. **Scripts de Automação:** Criar rotinas, para desligamento automático, redimensionamento e execução de tarefas recorrentes. **Auto-Scaling Dinâmico:** Utilizar recursos que ajustem a capacidade com base em métricas de uso em tempo real.		

Recurso	Descrição	Orientações de Avaliação de Disponibilidade e	Benefícios e impactos	Links de referência
Máquinas Virtuais	**Instâncias Spot - Otimização de Custos em Computação**: Aproveitamento de instâncias spot para reduzir custos em computação na nuvem, aproveitando preços mais baixos oferecidos por capacidade computacional não utilizada temporariamente.	**Preços**: Monitorar constantemente a disponibilidade e os preços das instâncias spot em diferentes regiões, implementando alertas e estratégias de fallback para garantir a continuidade dos serviços durante interrupções. Otimizar o uso com foco em custo-benefício, incluindo políticas de auto-scaling que respeitem os limites de orçamento e a performance necessária. **Testes Regulares**: Realizar testes periódicos para assegurar que a infraestrutura consiga lidar com interrupções inesperadas das instâncias spot.	Redução significativa nos custos de processamento Maior flexibilidade no gerenciamento de recursos computacionais Capacidade de escalar operações de maneira econômica.	AWS Spot Instances https://aws.amazon.com/pt/ec2/spot/ Azure Spot Virtual Machines https://azure.microsoft.com/pt-br/products/virtual-machines/spot Google Cloud Spot VMs https://cloud.google.com/spot-vms?hl=pt-br
Máquinas Virtuais	**Escolha da Máquina Virtual**: Escolher o tipo e tamanho de instância mais adequado à carga de trabalho, visando performance e custo-benefício.	**Análise de Utilização:** Monitorar continuamente a **Desempenho** e **Custos:** Conduzir análises periódicas do **Ajustes Regulares:** Revaliar e ajustar as estratégias a conforme necessário para **Comparação de Tipos:** Testar diferentes tipos de instâncias para encontrar o mais eficiente. **Auditorias Técnicas:** Realizar levantamentos periódicos sobre uso e desempenho. **Ferramentas Automatizadas:** Implementar soluções de monitoramento e escalonamento automático. **Revisão de Políticas:** Atualizar diretrizes de uso conforme as melhores práticas e mudanças na carga.	Otimização de custos operacionais Aumento da eficiência de recursos Desempenho aprimorado Redução de desperdício Flexibilidade e escalabilidade	AWS Instance Types https://aws.amazon.com/pt/ec2/instance-types/ Máquinas virtuais no Azure https://learn.microsoft.com/pt-br/azure/virtual-machines/ Google Compute Engine Machine Types https://cloud.google.com/compute/docs/machine-types?hl=pt-br
		Scripts de Automação: Criar rotinas para desligamento automático, redimensionamento e execução de tarefas recorrentes. **Auto-Scaling Dinâmico:** Utilizar recursos que ajustem a capacidade com base em métricas de uso em tempo real.		

Recurso	Descrição	Revisão e Otimização de Orientações	Benefícios e impactos	Links de referência
Rede	**Saída de rede:** Implementação gestão estratégica de Data-Out para reduzir os custos e otimizar o tráfego de dados que saem da nuvem.	**Configurações de Rede**: Analisar e otimizar as configurações de rede e acordos de nível de serviço para minimizar transferências desnecessárias de dados e seus custos associados. Estudar os padrões de tráfego e identificar fluxos de dados. **Implementação de Compressão de Dados**: Utilizar técnicas de compressão e caching para reduzir a quantidade de dados transferidos, melhorando a eficiência. **Configuração de Tarifas e Condições**: Estabelecer as melhores tarifas e condições de transferência de dados com provedores de nuvem para maximizar a economia.	Redução de custos de transferência de dados Melhor aproveitamento da largura de banda e aumento da eficiência operacional.	AWS Data Transfer https://docs.aws.amazon.com/pt_br/cur/latest/userguide/cur-data-transfers-charges.html Azure Data Transfer Pricing https://cloudmonitor.ai/2021/08/azure-data-transfer-costs-everything-you-need-to-know/ Network Pricing Google Cloud https://cloud.google.com/vpc/network-pricing?hl=pt_br
Máquinas Virtuais	**Escolha da Máquina Virtual**: Escolher o tipo e tamanho de instância mais adequado à carga de trabalho, visando performance e custo-benefício.	**Análise de Utilização:** Monitorar continuamente o consumo de CPU, memória e armazenamento. **Ajustes Regulares:** Empregar ferramentas de análise para identificar oportunidades de otimização. **Comparação de Tipos:** Testar diferentes tipos de instâncias para encontrar o mais eficiente. **Auditorias Técnicas:** Realizar levantamentos periódicos sobre uso e desempenho. **Ferramentas Automatizadas:** Implementar soluções de monitoramento e escalonamento automático. **Revisão de Políticas:** Atualizar diretrizes de uso conforme as melhores práticas e mudanças na carga.	Otimização de custos operacionais Aumento da eficiência de recursos Desempenho aprimorado Redução de desperdício Flexibilidade e escalabilidade	AWS Instance Types https://aws.amazon.com/pt/ec2/instance-types/ Máquinas virtuais no Azure https://learn.microsoft.com/pt-br/azure/virtual-machines/ Google Compute Engine Machine Types https://cloud.google.com/compute/docs/machine-types?hl=pt-br
		Scripts de Automação: Criar rotinas para desligamento automático, redimensionamento e execução de tarefas recorrentes. **Auto-Scaling Dinâmico:** Utilizar recursos que ajustem a capacidade com base em métricas de uso em tempo real.		

Recurso	Descrição	Orientações de Implementação	Benefícios e impactos	Links de referência
Rede	**Balanceamento de Carga**: distribuição eficaz do tráfego de rede entre várias instâncias para otimizar o desempenho.	**Balanceadores de Carga**: Configurar balanceadores de carga com políticas de gerenciamento de tráfego de entrada, garantindo alta disponibilidade e ajustando a capacidade conforme a demanda. **Testes de Estresse Periódicos**: Realizar testes de estresse regularmente para avaliar a performance e a resiliência da infraestrutura sob carga intensa. **Implementação de Auto-Recovery**: Estabelecer mecanismos de auto-recovery para restaurar automaticamente os serviços em caso de falhas.	Melhoria da disponibilidade do serviço; Redução de pontos únicos de falha; Distribuição eficiente de recursos e otimização de desempenho.	AWS Elastic Load Balancing https://aws.amazon.com/pt/elasticloadbalancing/ Azure Load Balancer https://learn.microsoft.com/pt-br/azure/load-balancer/load-balancer-overview Google Cloud Load Balancing https://cloud.google.com/load-balancing?hl=pt_br
Máquinas Virtuais	**Uso de recursos ou sistema de VPN** (Virtual Private Network): solução de comunicação entre redes de trabalho, visando infraestrutura pública com aplicação de mecanismos de segura em todo tráfego.	**Análise de Utilização:** Monitorar continuamente o consumo de CPU, memória e armazenamento. **Ajustes Regulares:** Revisar e ajustar instâncias conforme a demanda. **Escolha da Máquina Virtual**: Escolher o tipo e tamanho de instância mais adequado à carga de trabalho, visando performance e custo-benefício. **Uso de Rede Virtual Segura**: Estabelecer uma rede virtual segura que utilize criptografia. **Ferramentas Automatizadas:** Implementar soluções de monitoramento e escalonamento automático. **Revisão de Políticas:** Atualizar diretrizes de uso conforme as melhores práticas e mudanças na carga.	Otimização de custos operacionais Aumento da eficiência Acesso seguro de recursos Controle de acesso Desempenho aprimorado Prevenção de ataques Redução de desperdício Flexibilidade e escalabilidade	VPN AWS https://docs.aws.amazon.com/pt_br/whitepapers/latest/building-scalable-secure-multi-vpc-network-infrastructure/vpn.html Máquinas virtuais no Azure https://learn.microsoft.com/pt-br/azure/virtual-machines/ VPN Google Cloud https://cloud.google.com/network-connectivity/docs/vpn/concepts/overview?hl=pt-br
		Gerenciamento de Acessos: Adotar uma ferramenta que gerencia acessos de clientes de forma segura. **Comparação de Tipos:** Testar diferentes tipos de instâncias para encontrar a mais eficiente. **Auditorias Técnicas:** Realizar levantamentos periódicos sobre uso e desempenho. **Scripts de Automação:** Criar rotinas para desligamento automático, redimensionamento e execução de tarefas recorrentes. **Auto-Scaling Dinâmico:** Utilizar recursos que ajustem a capacidade com base em métricas de uso em tempo real.		AWS Instance Types https://aws.amazon.com/pt/ec2/instance-types/ Máquinas virtuais no Azure https://learn.microsoft.com/pt-br/azure/virtual-machines/point-to-site-about Google Compute Engine Machine Types https://cloud.google.com/compute/docs/machine-types?hl=pt-br

Recurso	Descrição	Análise de Tipos de Orientações	Benefícios e impactos	Links de referência
Armazenamento	Uso de sistemas modernos de armazenamento, compreendendo o tipo correto.	**Armazenamento**: Avaliar os melhores tipos de armazenamento que atendam à demanda com o menor custo possível. **Comparação de Opções de Armazenamento**: Analisar o uso entre EBS, Armazenamento de Instância, EFS e S3 para determinar a solução mais adequada. **Viabilidade do Uso de EBS**: Avaliar a viabilidade do uso de EBS nas instâncias, considerando desempenho e custo. **Priorizar o Uso do GP3**: Implementar, como prioridade, o uso de volumes GP3 para otimização de custo e desempenho.	Aumento no desempenho operacional Custo otimizado, dependendo do contrato Eficiência na persistência dos dados e na política de Backup	rmazenamento para o dispositivo raiz AWS https://docs.aws.amazon.com/pt_br/AWSEC2/latest/UserGuide/storage_ebs.html Amazon Elastic Block Store https://docs.aws.amazon.com/pt_br/ebs/latest/userguide/what-is-ebs.html Armazenamento de Arquivos do Azure https://azure.microsoft.com/pt-br/products/storage#Overview Google Cloud Disk https://cloud.google.com/compute/docs/disks?hl=pt-br Google Cloud Storage https://cloud.google.com/storage?hl=pt_br
Máquinas Virtuais	**Escolha da Máquina Virtual**: Escolher o tipo e tamanho de instância mais adequado à carga de trabalho, visando performance e custo-benefício.	**Análise de Utilização:** Monitorar continuamente o consumo de CPU, memória e armazenamento. **Ajustes Regulares:** Reavaliar e ajustar instâncias conforme a variação da demanda. **Comparação de Tipos:** Testar diferentes tipos de instâncias para encontrar o mais eficiente. **Auditorias Técnicas:** Realizar levantamentos periódicos sobre uso e desempenho. **Ferramentas Automatizadas:** Implementar soluções de monitoramento e escalonamento automático. **Revisão de Políticas:** Atualizar diretrizes de uso conforme as melhores práticas e mudanças na carga.	Otimização de custos operacionais Aumento da eficiência de recursos Desempenho aprimorado Redução de desperdício Flexibilidade e escalabilidade	AWS Instance Types https://aws.amazon.com/pt/ec2/instance-types/ Máquinas virtuais no Azure https://learn.microsoft.com/pt-br/azure/virtual-machines/ Google Compute Engine Machine Types https://cloud.google.com/compute/docs/machine-types?hl=pt-br
		Scripts de Automação: Criar rotinas para desligamento automático, redimensionamento e execução de tarefas recorrentes. **Auto-Scaling Dinâmico:** Utilizar recursos que ajustem a capacidade com base em métricas de uso em tempo real.		

Recurso	Descrição	Orientações	Benefícios e impactos	Links de referência
Monitoramento e Observabilidade	Sistemas de monitoramento das aplicações para prever e resolver problemas que afetem os serviços proativamente, assim como visualização do status e estatísticas e uma base de dados dos logs de eventos.	**Sistemas de Monitoramento**: Utilizar sistemas de monitoramento para coleta contínua e em tempo real de dados sobre desempenho, saúde e eventos das aplicações, empregando ferramentas com painéis de fácil visualização. **Configuração de Alertas**: Estabelecer alertas para eventos críticos, garantindo respostas rápidas a incidentes. **Análise de Desempenho**: Realizar análises regulares de desempenho para identificar áreas de melhoria. **Adaptações Baseadas em Insights**: Implementar adaptações e otimizações com base nos insights obtidos a partir dos dados coletados.	Prevenção de atraso de respostas e indisponibilidade Otimização de desempenho Decisões baseadas em dados Redução de custos de manutenção	AWS Cloudwatch https://aws.amazon.com/pt/cloudwatch/ AWS Cloudtrail https://aws.amazon.com/pt/cloudtrail/ Azure Monitor https://azure.microsoft.com/pt-br/products/monitor
Máquinas Virtuais	**Escolha da Máquina Virtual**: Escolher o tipo e tamanho de instância mais adequado à carga de trabalho, visando performance e custo-benefício.	**Análise de Utilização**: Monitorar continuamente o consumo de CPU, memória, e armazenamento. **Ajustes Regulares**: Reavaliar e ajustar instâncias conforme a variação da demanda. **Comparação de Tipos**: Testar diferentes tipos de instâncias para encontrar o mais eficiente. **Auditorias Técnicas**: Realizar levantamentos periódicos sobre uso e desempenho.	Melhor planejamento de capacidade, aumento da satisfação do usuário Fonte de dados para otimização de custos Aumento da eficiência de recursos Desempenho aprimorado Redução de desperdício Flexibilidade e escalabilidade	Google Cloud Monitoring https://cloud.google.com/monitoring?hl=pt-br AWS Instance Types https://aws.amazon.com/pt/ec2/instance-types/ Máquinas virtuais no Azure https://learn.microsoft.com/pt-br/azure/virtual-machines/ Google Compute Engine Machine Types https://cloud.google.com/compute/docs/machine-types?hl=pt-br
Segurança de Web Firewall	Sistema de proteção contra explorações e ataques Web que podem afetar a disponibilidade, comprometer a segurança ou consumir recursos.	**Treinamento da Equipe Técnica**: Proporcionar treinamento contínuo para a equipe técnica, garantindo que estejam atualizados sobre as melhores práticas. **Documentação de Incidentes**: Documentar incidentes e revisar métricas. **Ferramentas Automatizadas: Implementação de Regras de Segurança**: Estabelecer regras de boas práticas de segurança que controlem o tráfego de entrada da rede e bloqueiem padrões de ataque conhecidos. **Revisão de Políticas:** Atualizar diretrizes de uso conforme as melhores práticas e mudanças na carga. **Análise de Comportamento de Acesso**: Monitorar e analisar os comportamentos e registros de acesso e execução de tarefas recorrentes. identificar atividades suspeitas e aprimorar a segurança. **Scripts de Automação:** Criar rotinas para desligamento automático, redimensionamento de aplicações para identificar. **Auto-Scaling Dinâmico:** Utilizar recursos que ajustem a capacidade com base em métricas de uso em tempo real.	Aumento na confiabilidade, integridade e disponibilidade as aplicações e seus recursos, garantindo maior eficiência na gestão do ambiente.	AWS WAF https://aws.amazon.com/pt/waf/ Azure WAF https://azure.microsoft.com/pt-br/products/web-application-firewall/ Google Cloud Armor https://cloud.google.com/security/products/armor?hl=pt_br

Recurso	Descrição	Orientações de Configuração	Benefícios e impactos	Links de referência
Serviço de Backup	É um serviço gerenciado que facilita a centralização e automatização da proteção de dados de qualquer serviço utilizado no ambiente.	**Centralizada de Políticas**: Implementar uma configuração centralizada de políticas exclusivas e de cofre dedicado, permitindo o monitoramento e a automação de tarefas programadas sem a necessidade de scripts personalizados ou processos manuais. **Criptografia Nativa e Integrada**: Disponibilizar recursos de criptografia nativos e integrados para garantir a segurança dos dados de forma eficiente e prática.	Programação de rotinas de backup e definição de período de retenção Habilitação de backup contínuo e recuperação pontual (PITR – Point-In-Time Recovery)	AWS Backup https://docs.aws.amazon.com/pt_br/aws-backup/latest/devguide/whatisbackup.html Azure Backup https://azure.microsoft.com/pt-br/products/backup Google Backup and Disaster Recovery https://cloud.google.com/backup-disaster-recovery?hl=pt-BR
Máquinas Virtuais	**Escolha da Máquina Virtual**: Escolher o tipo e tamanho de instância mais adequado à carga de trabalho, visando performance e custo-benefício.	**Análise de Utilização:** Monitorar continuamente o consumo de CPU, memória e armazenamento. **Ajustes Regulares:** Reavaliar e ajustar instâncias conforme a variação da demanda. **Comparação de Tipos:** Testar diferentes tipos de instâncias para encontrar o mais eficiente. **Auditorias Técnicas:** Realizar levantamentos periódicos sobre uso e desempenho. **Ferramentas Automatizadas:** Implementar soluções de monitoramento e escalonamento automático. **Revisão de Políticas:** Atualizar diretrizes de uso conforme as melhores práticas e mudanças na carga.	Otimização de custos operacionais Aumento da eficiência de recursos Desempenho aprimorado Redução de desperdício Flexibilidade e escalabilidade	AWS Instance Types https://aws.amazon.com/pt/ec2/instance-types/ Máquinas virtuais no Azure https://learn.microsoft.com/pt-br/azure/virtual-machines/ Google Compute Engine Machine Types https://cloud.google.com/compute/docs/machine-types?hl=pt-br
		Scripts de Automação: Criar rotinas para desligamento automático, redimensionamento e execução de tarefas recorrentes. **Auto-Scaling Dinâmico:** Utilizar recursos que ajustem a capacidade com base em métricas de uso em tempo real.		

Recurso	Descrição	Orientações	Benefícios e impactos	Links de referência
Containers	Containers são ferramentas que permitem virtualização em nível de sistema operacional. Eles operam de forma independente entre si. Um container encapsula seu código-fonte, bibliotecas, frameworks, entre outros artefatos dentro de seu próprio "ambiente". Essa independência permite que os containers executem suas cargas de trabalho independentemente do Sistema Operacional hospedeiro ou versão de Kernel.	**Imagens Leves e Seguras**: Utilizar imagens leves, catalogadas e seguras, como Alpine Linux, para otimizar o desempenho e a segurança. **Gerenciamento de Código**: Usar um sistema de controle de versão distribuído, como Git, para o gerenciamento de código-fonte e bibliotecas. **Armazenamento Seguro de Configurações**: Armazenar configurações e senhas em um cofre de segredos para proteger informações sensíveis. **Automatização de Implantação**: Automatizar o processo de implantação utilizando pipelines de CI/CD para melhorar a eficiência e a consistência. **Análise de Utilização:** Monitorar continuamente o consumo de CPU, memória e armazenamento. **Varredura de Vulnerabilidades Regulares:** Executar varreduras de vulnerabilidades e detecção de erros em containers com scanners. **Comparação de Tipos:** Testar diferentes tipos de instâncias para encontrar o mais eficiente. **Princípio do Menor Privilégio**: Aplicar o princípio do menor privilégio na execução de processos, evitando o uso de containers com privilégios elevados. **Remoção de Bibliotecas Desnecessárias**: Remover bibliotecas desnecessárias para reduzir a superfície de ataque.	Portabilidade garantida pela possibilidade de migração entre diferentes plataformas de nuvem ou ambientes on-premise, sem demanda por esforços excessivos Melhoria nos processos de manutenção e evolução de sistemas Replicação padronizada em múltiplos ambientes (produção, homologação e desenvolvimento) Otimização de custos operacionais Otimização da utilização de recursos Aumento da eficiência na alocação de recursos Desempenho aprimorado Elevação do nível de segurança por meio do isolamento entre containers. Redução de desperdício Flexibilidade e escalabilidade	Amazon Elastic Container Service - ECS https://aws.amazon.com/pt/ecs/ Azure Container https://azure.microsoft.com/pt-br/products/category/containers Google Container https://cloud.google.com/learn/what-are-containers?hl=pt-br
Máquinas Virtuais	**Escolha da Máquina Virtual**: Escolher o tipo e tamanho de instância mais adequado à carga de trabalho, visando performance e custo-benefício.	**Auditorias Técnicas:** Realizar levantamentos periódicos sobre o uso e desempenho. **Ferramentas Automatizadas:** Implementar soluções de monitoramento automatizadas para identificar ociosidade. **Revisão de Políticas:** Atualizar políticas de uso conforme necessário, com base em melhores práticas e mudanças na carga. **Scripts de Automação:** Criar rotinas para desligamento automático, redimensionamento e execução de tarefas recorrentes. **Auto-Scaling Dinâmico:** Utilizar recursos que ajustem a capacidade com base em métricas de uso em tempo real.	Aumento da eficiência na alocação de recursos Desempenho aprimorado Redução de desperdício Flexibilidade e escalabilidade	AWS Instance Types https://aws.amazon.com/pt/ec2/instance-types/ Máquinas virtuais no Azure https://learn.microsoft.com/pt-br/azure/virtual-machines/ Google Compute Engine Machine Types https://cloud.google.com/compute/docs/machine-types?hl=pt-br

2.2 PaaS: Recursos de Plataforma como Serviço

Recurso	Descrição	Orientações	Benefícios e impactos	Links de Referência

Recurso	Descrição	Orientações	Benefícios e impactos	Links de Referência
Sistema otimizadores de banco de dados	Sistema de gerenciamento voltado à automação e otimização dos principais bancos de dados disponíveis no mercado.	Implementação de ferramenta para gerenciamento dos bancos de dados e escalonamento conforme a demanda.	Automação das operações de banco de dados relacionais, abrangendo tarefas genéricas de gerenciamento, provisionamento, configuração, realização de backups e aplicação de patches.	AWS RDS https://aws.amazon.com/pt/rds/ Azure Data Bases https://azure.microsoft.com/pt-br/solutions/databases/ Google Cloud Data Bases https://cloud.google.com/products/databases?hl=pt-br
Orquestração de containers	Gerenciamento da vida útil de containers em ambientes de grande escala, abrangendo automação de implantações, escalonamento, configuração de rede e administração do estado dos containers.	Seleção do provedor de cloud com base em critérios como custo, suporte e localização geográfica, entre outros; Implementação inicial por meio de projetos piloto, com garantia de maior precisão e realização de ajustes antes da expansão em larga escala; Adoção de práticas robustas de segurança para proteção das aplicações e dos dados; Reconhecimento da complexidade na orquestração de containers, com exigência de conhecimento especializado.	Facilidade no escalonamento de aplicações para atendimento a variações na demanda; Garantia de alta disponibilidade das aplicações por meio de estratégias de redundância e failover.	AWS Elastic Kubernetes Service (EKS) https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html Azure Kubernetes Service (AKS) https://learn.microsoft.com/en-us/azure/aks/ Google Kubernetes Engine (GKE) https://cloud.google.com/kubernetes-engine

2.3 SaaS: Recursos de Software como como Serviço

Recurso	Descrição	Orientações	Benefícios e impactos	Links de referência

Recurso	Descrição	Orientações	Benefícios e impactos	Links de referência
Gestão de segurança	Serviço de segurança que proporciona uma visão unificada e centralizada da infraestrutura, oferecendo insights acionáveis sobre a situação de segurança e facilitando a identificação, priorização e mitigação de riscos.	**Análise Proativa:** Implementação de processos para revisão regular das recomendações do sistema, com priorização de ações baseada em criticidade e impacto potencial na infraestrutura. **Documentação e Padronização:** Registro dos fluxos de segurança, abrangendo políticas de resposta a incidentes, mecanismos de mitigação e configurações de alerta, para garantia de consistência e reprodutibilidade. **Automação e Resiliência:** Adoção de ferramentas de automação para aplicação de correções de segurança e configuração de alertas em tempo real, com redução da dependência de intervenção manual e ampliação da capacidade de resposta a ameaças.	**Simplificação da Coleta de Dados de Segurança:** Centralização das informações de segurança, com eliminação da complexidade no gerenciamento de dados dispersos em múltiplas fontes e facilitação de análise ágil e eficaz da postura de segurança da infraestrutura. **Visão Unificada de Ameaças e Vulnerabilidades:** Fornecimento de interface consolidada para monitoramento, priorização de riscos e resposta a incidentes, com diminuição do tempo e esforço na análise manual de dados heterogêneos. **Suporte à Tomada de Decisão:** Inclusão de análise detalhada das recomendações geradas pelo sistema, com promoção de discussões colaborativas entre a equipe técnica para definição das melhores ações. **Construção de uma Base de Conhecimento:** Criação e manutenção de repositório de melhores práticas e lições aprendidas em segurança na nuvem, com sincronização às recomendações do serviço, para funcionamento como referência contínua e evolutiva.	AWS Security Hub https://aws.amazon.com/pt/security-hub/ Azure Defender for Cloud https://azure.microsoft.com/pt-br/products/defender-for-cloud/ Cloud Security Command Center Google Cloud https://cloud.google.com/security-command-center

Recurso	Descrição	Orientações	Benefícios e impactos	Links de referência
Rede de Distribuição de Conteúdo (CDN)	Consistem em uma infraestrutura distribuída que otimiza a entrega de conteúdo digital, armazenando cópias em servidores estrategicamente localizados (edge servers) próximos aos usuários finais. Esse modelo reduz a latência, melhora a eficiência na entrega e alivia a carga sobre os servidores de origem..	**Uso em Datacenters Fora do Brasil:** Avaliação da viabilidade e das vantagens, tanto técnicas quanto financeiras, na hospedagem de serviços em datacenters internacionais, com consideração de fatores como proximidade geográfica dos usuários, custos operacionais e desempenho de rede. **Aplicações com Conteúdo Estático:** Análise da implementação de CDNs em cenários com forte dependência de conteúdo estático (imagens, vídeos, arquivos CSS, JavaScript), onde a distribuição eficiente possibilita maximização do desempenho e da experiência do usuário.	**Melhoria no Desempenho** Aceleração do carregamento de páginas e aplicações por meio do armazenamento em cache de conteúdo em locais próximos aos usuários, com redução do tempo de resposta. **Redução de Latência** Minimização dos atrasos na entrega de dados pela diminuição da distância física entre o usuário e o servidor responsável pelo fornecimento do conteúdo. **Economia de Largura de Banda** Redução da quantidade de dados transferidos diretamente pelo servidor de origem, com consequente diminuição do consumo de banda e dos custos operacionais. **Aumento de Disponibilidade e Resiliência** Elevação da tolerância a falhas e da redundância pela distribuição do tráfego entre múltiplos servidores replicados, com garantia de alta disponibilidade mesmo em cenários de falhas regionais. **Reforço da Segurança** Oferta de camadas adicionais de proteção, incluindo mitigação de ataques DDoS, filtragem de tráfego malicioso e suporte a certificados SSL, conforme as funcionalidades da CDN selecionada.	Amazon CloudFront https://aws.amazon.com/cloudfront/ Azure CDN https://azure.microsoft.com/en-us/services/cdn/ Google Cloud CDN https://cloud.google.com/cdn
Gestão de Credenciais	Gestão de segredos de maneira centralizada.	Gerenciamento dos segredos com garantia de acesso seguro para todos os usuários da equipe; Criptografia e auditoria dos segredos, abrangendo credenciais de sistemas operacionais, bancos de dados e chaves de API.	Otimização da gestão das credenciais. Redução dos riscos de vazamento de informações críticas.	AWS Secrets Manager https://aws.amazon.com/pt/secrets-manager/ Azure Key Vault https://azure.microsoft.com/pt-br/products/key-vault/ Google Cloud Key Management https://cloud.google.com/security/products/security-key-management?hl=pt-BR

Recurso	Descrição	Orientações	Benefícios e impactos	Links de referência
Gestão de identidade e Acesso	A gestão de identidade constitui elemento essencial para a segurança e a administração de infraestrutura em nuvem. Envolve a utilização de tecnologias e a implementação de políticas voltadas à garantia de acesso adequado aos recursos de hardware e software pelas pessoas corretas. Esse processo tem como finalidade a criação de ambiente seguro e eficiente para os recursos de nuvem, com proteção contra acessos não autorizados.	Definição de políticas claras para acesso aos recursos. Identificação de circunstâncias relacionadas à concessão, revisão e revogação de acessos. Revisão periódica dos acessos autorizados, com foco na detecção de usuários fora da organização e permissões desnecessárias. Utilização de políticas robustas de senha, com inclusão de números, letras maiúsculas e minúsculas, caracteres especiais e estabelecimento de limite mínimo de caracteres. Determinação de prazo para expiração de senhas. Adoção de métodos avançados de autenticação, por meio da implementação de Autenticação Multi-Fator (MFA). Aplicação do princípio do menor privilégio (POLP), com concessão de acesso e permissões a contas, aplicativos e dispositivos apenas para execução de suas funções; Emprego de ferramenta centralizada para controle de identidades, com minimização do esforço no gerenciamento das contas da organização; Implementação de auditoria e monitoramento para identificação de atividades suspeitas ou não autorizadas.	Melhoria da segurança. Flexibilidade na gestão de identidades e acessos.	AWS Identity and Access Management (IAM) https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html Azure Active Directory (Azure AD) https://azure.microsoft.com/en-us/services/active-directory/ Google Cloud Identity & Access Management (IAM) https://cloud.google.com/iam

Recurso	Descrição	Orientações	Benefícios e impactos	Links de referência
Gestão de Custos	Auxílio na configuração de faturamento, organização e planejamento de custos.	Orientação à equipe para otimização do uso dos recursos geradores de gastos, com realização de sanitização do ambiente (exclusão de itens em desuso ou excessivos). Monitoramento contínuo dos recursos utilizados. Aplicação de políticas de gerenciamento de custos, com análise e apuração da utilização de serviços gerenciados.	Previsibilidade financeira. Utilização eficiente de recursos. Ampliação da transparência e visibilidade. Criação de cultura de eficiência, com promoção da prática de gestão e controle de custos para cultivo de responsabilidade financeira	AWS Cost Explorer https://aws.amazon.com/pt/aws-cost-management/aws-cost-explorer/ Azure Cost Management https://azure.microsoft.com/en-us/products/cost-management Gerenciamento de custos Google Cloud https://cloud.google.com/cost-management?hl=pt_br
Gateways NAT	Serviço que facilita a conexão de instâncias em sub-redes privadas à Internet, com impedimento de recebimento de conexões de entrada não solicitadas provenientes da Internet. Disponibilização de dois tipos de gateways NAT: público e privado.	Configuração de gateway de conversão de endereços de rede (NAT), com permissão para conexão de instâncias em sub-rede privada a serviços externos à VPC e prevenção de iniciação de conexões por esses serviços externos às instâncias.	Contribuição de gestão eficaz de acessos para segurança da aplicação e utilização eficiente de recursos, com prevenção de ataques cibernéticos.	Gateways NAT AWS https://docs.aws.amazon.com/pt_br/vpc/latest/userguide/vpc-nat-gateway.html Gateways NAT Azure https://learn.microsoft.com/pt-br/azure/nat-gateway/nat-gateway-resource?source=recommendations Cloud NAT Google Cloud https://cloud.google.com/nat/docs/overview?hl=pt-br