CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE LINS PROF. ANTONIO SEABRA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANDRÉ PARRA DOURADO

PROVISIONAMENTO DE INFRAESTRUTURA EM CLOUD POR MEIO DE FERRAMENTA IAC

LINS/SP
1º SEMESTRE/2023

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE LINS PROF. ANTONIO SEABRA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANDRÉ PARRA DOURADO

PROVISIONAMENTO DE INFRAESTRUTURA EM CLOUD POR MEIO DE FERRAMENTA IAC

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Lins para a obtenção do título de Tecnólogo(a) em Análise e Desenvolvimento de Sistemas

Orientador: Prof. Me. Felipe Maciel

LINS/SP
1º SEMESTRE/2023

Parra Dourado, André.

P258p Provisionamento de infraestrutura em cloud por meio de ferramenta IaC / André Parra Dourado. – Lins, 2023.

63f

Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – Faculdade de Tecnologia de Lins Professor Antonio Seabra: Lins, 2023.

Orientador(a): Me. Felipe Maciel Rodrigues.

1. Cloud Computing. 2. Cloud Native. 3. Infrastructure as Code. 4. Google Cloud Platform. 5. Terraform. I. Maciel Rodrigues, Felipe. II. Faculdade de Tecnologia de Lins Professor Antonio Seabra. III. Título.

CDD 004.21

ANDRÉ PARRA DOURADO

PROVISIONAMENTO DE INFRAESTRUTURA EM CLOUD POR MEIO DE FERRAMENTA IAC

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Lins como parte dos requisitos para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas sob orientação do prof. me. Felipe Maciel

Data de aprovação: 14/06/2023

Orientador: Me. Felipe Maciel Rodrigues

Examinador 1: Me. Adriano Bezerra

Examinador 2: Me. Júlio Fernando Lieira

Dedico este trabalho aos meus amados pais, Edgard e Mônica, e a todos os meus amados familiares.

André Parra Dourado

AGRADECIMENTOS

Aos meus amados pais por tudo. O amor, a compreensão e a educação recebida deles me tornou e me torna uma pessoa melhor.

A cada professor que fez parte desta trajetória, por compartilhar seus conhecimentos e esclarecer as dúvidas trazidas pelos alunos em sala de aula, contribuindo com nosso crescimento pessoal e profissional.

Aos colegas de sala Vinicius e Vitor, por compartilharem o caminho e os conhecimentos desde o momento em que ingressei no curso.

Em especial, aos professores Lourenço, Tuca e Fernanda por cada discussão que envolveu assuntos mais reflexivos.

Aos professores, orientadores, e amigos Rafael e Felipe que me apoiaram ao longo da faculdade e da confecção deste TCC.

Por fim, a todos os amigos que fizeram desta trajetória mais leve e divertida.

André Parra Dourado

RESUMO

Em um passado recente, as empresas que tinham interesse em armazenar, processar e extrair informações de seus dados, necessitavam se planejar para estruturar fisicamente *data centers* ou alugar *co-locations/housing* para viabilizar os seus propósitos. Ambas as maneiras citadas (criação de um *data center* ou locação de *co-locations/housing*) são trabalhosas, burocráticas e limitadas em comparação às possibilidades que existem no cenário atual da *cloud computing*. Dada a elasticidade e escalabilidade, os recursos da *cloud computing* têm incentivado as empresas para o processo de migração das instalações *on-premises* para a *cloud*. Este aquecimento da área de Tecnologia da Informação (T.I) resulta em diversos projetos onde frequentemente a estruturação de uma infraestrutura em *cloud* é crucial. Neste contexto, surgem ferramentas nomeadas de *Infrastructure as Code* (IaC), do português, infraestrutura como código, cujo propósito é auxiliar desde a concepção da infraestrutura na *cloud* até a sua manutenção. Este projeto tem como objetivo por meio de ferramenta de IaC, o Terraform, provisionar uma infraestrutura na Google Cloud Platform (GCP) dois ambientes, produção e gerência. O objetivo da infraestrutura provisionada é fornecer um ambiente de trabalho flexível, que suporte aplicações legadas e *cloud native*. Utilizando-se dos conceitos IaaS, PaaS, DBaaS e SaaS, foram implantados nos ambientes deste projeto: *clusters* Kubernetes não gerenciado e autogerenciado, *cluster* Dataproc, máquina virtual com servidor Apache disponibilizando um website, máquina virtual com servidor Nginx disponibilizando uma aplicação Zabbix que possui banco de dados relacional em uma instância apartada, por fim um *website* estático com *load balancer*.

Palavras–chave: *Cloud Computing. Cloud Native. Infrastructure as Code*. Google Cloud Platform. Terraform.

ABSTRACT

In the recent past, companies that were interested in storing, processing and extracting information from their data needed to plan to physically structure data centers or rent co-locations/housing to make their purposes viable. Both ways mentioned (creating a data center or leasing co-locations/housing) are laborious, bureaucratic and limited compared to the possibilities that exist in the current scenario of cloud computing. Given the elasticity and scalability, cloud computing resources have encouraged companies to migrate from on-premises installations to the cloud. This warming up of the Information Technology (IT) area results in several projects where often the structuring of a cloud infrastructure is crucial. In this context, tools named Infrastructure as Code (IaC) appear, from Portuguese, infraestrutura como código, whose purpose is to assist from the design of the infrastructure in the cloud to its maintenance. This project aims, through the IaC tool, Terraform, to provision an infrastructure on Google Cloud Platform (GCP) for two environments, production and management. The purpose of the provisioned infrastructure is to provide a flexible working environment that supports both legacy and cloud native applications. Using the IaaS, PaaS, DBaaS and SaaS concepts, the following were implemented in the environments of this project: unmanaged and self-managed Kubernetes clusters, Dataproc cluster, virtual machine with Apache server providing a website, virtual machine with Nginx server providing a Zabbix application that has relational database in a separate instance, finally a static website with load balancer.

Keywords: Cloud Computing. Cloud Native. Infrastructure as Code. Google Cloud Platform. Terraform.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AWS – Amazon Web Services

Blob – Binary Large Object

CaaS – Container as a Service

CLI – Command-line Interface

CRM - Customer Relationship Management

DBaaS – Database as a Service

DKIM – DomainKeys Identified Mail

DMARC – Domain-based Message Authentication Reporting and Conformance

DNS – Domain Name System

FaaS – Function as a Service

GCE – Google Compute Engine

GCP – Google Cloud Platform

GCS – Google Cloud Storage

GKE – Google Kubernetes Engine

GUI – Graphical User Interface

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

IaC – Infrastructure as Code

IaaS – Infrastructure as a Service

IAM – Identity and Access Management

IP – Internet Protocol

LB – Load Balancer

MX – Mail Exchange

PaaS – Platform as a Service

SaaS – Software as a Service

SDK – Software Development Kit

SGBD – Sistema de Gerenciamento de Banco de Dados

SPF – Sender Policy Framework

SSL – Secure Sockets Layer

Subnet – Subnetwork

TF - Terraform

TI – Tecnologia da Informação

TLS – Transport Layer Security

UI – User Interface

VM – Virtual Machine

VPC – Virtual Private Cloud

URL – Uniform Resource Locator

WSL – Windows Subsystem for Linux

SUMÁRIO

1 INTRODUÇÃO

O mundo está em constante transformação e a tecnologia está mudando radicalmente a maneira como as empresas operam e interagem com seus clientes. A transformação digital refere-se à incorporação estratégica de tecnologias digitais em todos os aspectos de uma empresa, desde a infraestrutura até os processos internos e o envolvimento com os clientes. Essa mudança revolucionária permite que as empresas otimizem suas operações, aprimorem a eficiência, alcancem maior agilidade e proporcionem uma experiência excepcional aos clientes.

> A globalização nas últimas décadas colocou uma pressão crescente sobre as empresas para mudar. Isso exige que as empresas se integrem com eficiência para não apenas permanecerem vivas, mas também prosperarem em ambientes competitivos. A integração eficiente só pode ser alcançada por meio de processos digitais e ferramentas colaborativas (White, 2012). Sendo assim, a importância da transformação digital (DT) aumentou. A pesquisa enfatiza que o DT deve ser incluído nas perspectivas de negócios existentes, pois esse tópico aborda muito mais do que apenas mudanças tecnológicas (Bouncken et al., 2021) e afeta muitos ou todos os segmentos de negócios: a transformação de negócios bem-sucedida é alcançada explorando e explorando o que ela oferece para alcançar a agilidade organizacional (Hess et al., 2016). (White, Bouncken, Hess, 2012, 2021, 2016, apud Kraus et. al., 2023, p. 1)

Dentre as vantagens da transformação digital, pode-se citar a capacidade de coletar e analisar grandes volumes de dados. Neste contexto, as empresas podem aproveitar-se desses dados para obter percepções/ideias, tais como: identificar o comportamento do cliente, detectar padrões, antecipar tendências de mercado e tomar decisões estratégicas embasadas em informações assertivas. Isso permite que as empresas sejam mais ágeis em suas operações, respondendo rapidamente às demandas do mercado e oferecendo produtos e serviços personalizados.

Adicionalmente, a digitalização de processos e a automação também contribuem para a redução de custos operacionais e aumentam a eficiência, permitindo que as empresas sejam mais competitivas em escala global. No entanto, para que tudo isso seja possível, é preciso também aderir a novas tecnologias e a novos modelos de implementação de infraestruturas que tenham agilidade, escalabilidade e elasticidade para acompanhar as rápidas e agressivas variações de mercado.

Um ponto a se ressaltar é que as empresas precisam ter um local munido de tecnologias para que possam ter seus dados, serviços e aplicações digitais armazenados, processados, acessados e distribuídos. Esta infraestrutura é popularmente conhecida como *data center* e podem ser categorizadas em diferentes tipos. Segundo VMware (2023a): “Um *data center* é uma instalação física centralizada onde se encontram computadores corporativos, rede, armazenamento e outros equipamentos de TI que dão suporte às operações de negócios”.

Em relação as diferentes categorias de *data centers*, SUSE (2023) explica que a tecnologia e *software on-premises* é mantida dentro da instalação física de uma empresa, ou seja, em um *data center on-premises* toda a infraestrutura física é de responsabilidade desta empresa.

Outro modelo é o *colocation data center*, que Fortinet (2023) define como um *data center* que oferece a possibilidade de locação de espaço para hospedar seus servidores, ou seja, uma infraestrutura pronta onde pode-se aproveitar tudo que foi projetado e construído para este fim, envolvendo rede, segurança, refrigeração, manutenção entre outros aspectos.

Por fim, há o modelo mais atual, escalável e expansível que é a *cloud computing*, ou computação/infraestrutura em nuvem. Google (2023c) define que a *cloud computing* é a disponibilidade sob demanda dos recursos de computação como serviços na Internet eliminando a necessidade de as empresas adquirirem, configurarem ou gerenciarem a infraestrutura, além de gerar custos apenas pelos recursos que elas utilizarem, pelo tempo que utilizarem.

Neste último modelo, o foco é em aplicações enxutas e otimizadas para que seja consumido o mínimo possível, acarretando o menor custo possível, ou seja, com a *cloud computing* (ou apenas *cloud),* é possível usufruir das infraestruturas de empresas renomadas e consolidadas no mercado, tais como Google, Microsoft, Amazon, Oracle, etc. que atualmente são consideradas sinônimo de confiabilidade, disponibilidade, escalabilidade e elasticidade. As empresas que fornecem serviços de *cloud computing* são chamadas de *cloud providers.*

A confiabilidade e disponibilidade são provenientes das questões de construção estratégica (área, refrigeração, capacidade de expansão, fornecimento de eletricidade, segurança) e da manutenção por equipes altamente especializadas que monitoram os *data centers* 24/7 (24 horas por dia e 7 dias por semana). Por exemplo,

a grande parte das taxas de disponibilidade do Google Cloud Platform (GCP) ultrapassam os 99% ao ano.

Os termos escalabilidade e elasticidade, se referem à velocidade e a capacidade que a infraestrutura tem de se adequar as diversas oscilações de demanda de cada um de seus consumidores. Segundo Qi Network (2023): “A escalabilidade se refere a capacidade de ampliar de forma ilimitada o armazenamento de toda a infraestrutura, *softwares* e servidores das empresas em plataformas em nuvem, como o Google Cloud Platform”.

Em outras palavras, a escalabilidade refere-se ao poder que uma *cloud provider* tem de sustentar as infraestruturas de diversos clientes e ainda fornecer a possibilidade de expansão de recursos caso os clientes necessitem. Já VMware (2023b) explica que elasticidade é uma propriedade que uma infraestrutura em *cloud* tem de aumentar ou diminuir recursos (memória, cpu, armazenamento, etc) adaptando-se as demandas de um cliente.

As adaptações baseadas nas demandas visam agilidade, visto que a maioria das aplicações tem momentos de alta demanda e pouca demanda, como por exemplo, sites que realizam *black friday* - promoção anual amplamente divulgada pelas lojas onde recebem uma quantidade de acessos muito superior em relação aos dias “comuns”. Por meio da elasticidade, as empresas têm a seu dispor o potencial computacional para atender picos de demanda, consumindo apenas o necessário requisitado e, após isso, reduzir novamente a infraestrutura adequando-se ao consumido usualmente.

O objetivo deste trabalho é disponibilizar uma infraestrutura por meio de IaC o qual seja possível demonstrar as qualidades, como flexibilidade, agilidade, elasticidade, da infraestrutura em *cloud.* Ademais uma ferramenta IaC agrega o gerenciamento, versionamento e automatização, por meio da criação de ambientes que sirvam para hospedar tanto as aplicações convencionais quanto as aplicações *cloud native* e que sejam de fácil monitoramento.

2 CONCEITOS BÁSICOS

2.1 AMBIENTES DE TRABALHO

Nos dias atuais, em um cenário ideal e padronizado, as empresas têm suas demandas divididas em pelo menos quatro ambientes distintos: desenvolvimento *(development)*, homologação *(staging)*, produção *(production)* e gerência *(management)*.

Microsoft (2023b) específica que no ambiente de desenvolvimento é onde as alterações nas aplicações são realizadas. É um ambiente com poucas restrições, permitindo o desenvolvimento das aplicações e a realização exaustiva dos mais diversos testes envolvendo integrações e diferenças de versões entre os seus componentes e recursos.

O ambiente de homologação hospeda as aplicações desenvolvidas, no ambiente de desenvolvimento, para testes finais antes de ir para o ambiente de produção. Microsoft (2023b) explica que o ambiente de homologação visa ser uma cópia fiel do ambiente de produção. Este ambiente possibilita análises de comportamento da aplicação em conjunto com suas integrações simulando um cenário real. Esta visibilidade refinada pré-implantação indica ajustes e correções necessárias para que a implantação em ambiente de produção seja assertiva.

Segundo Microsoft (2023b) é no ambiente de produção que se encontram as aplicações que de fato estão sendo utilizadas pelos clientes das aplicações/serviços. Ou seja, é um ambiente restrito, monitorado e controlado em que o ideal é não ocorrer testes e intervenções reativas, e sim passar a maior parte do tempo disponível e sem interrupções visto que, nesse ambiente, segundos de lentidão/indisponibilidade costumam gerar grandes prejuízos.

Por fim, o ambiente de gerência é o ambiente em que se centraliza a parte principal de segurança, monitoramento e integração entre os outros ambientes. Neste projeto, o ambiente de gerência utiliza da shared VPC, em português, VPC compartilhada, para que isto seja possível. Segundo Google (2023e), a shared VPC permite implementar melhores práticas de segurança, auditoria e controle de acesso para projetos de uma organização.

Como citado anteriormente, este é o cenário ideal, mas que infelizmente, não é adotado na maioria dos casos. Muitas vezes são adotados apenas os ambientes “essenciais”, sendo eles gerência e produção, onde o ambiente de produção acaba incorporando as “funções” dos ambientes de desenvolvimento e homologação.

Neste trabalho é implantado os ambientes de produção e gerência, a fim de demonstrar as funcionalidades e possibilidades das ferramentas *Infrastructure as Code* (IaC, em português, ferramentas de infraestrutura como código).

2.2 INFRAESTRUTURAS

A infraestrutura de computação desempenha um papel fundamental na área da tecnologia da informação, provendo a base necessária para o funcionamento eficiente de sistemas e serviços computacionais. Como mencionado anteriormente, o avanço contínuo da tecnologia tem proporcionado diferentes tipos de infraestruturas de computação, cada um com suas características e aplicabilidades específicas. Nesta seção são conceituados os diferentes tipos de infraestruturas de computação explorando as suas principais características.

Vale ressaltar que no contexto atual, em que a *cloud computing*, a virtualização e outras tecnologias emergentes estão transformando a maneira como as organizações utilizam os recursos computacionais, tais informações são relevantes. Deste modo, é possível tomar decisões estratégicas em relação à escolha adequada para atender às necessidades de um projeto ou organização.

2.2.1 On-premises

O termo *on-premises* refere-se a infraestrutura computacional de uma empresa instalada localmente na área da própria empresa, sendo consumo e acesso restrito apenas a ela. Segundo Hughes L, Kasunic M, Sweeney D (2023, p. 1, tradução nossa) “Os *data centers on-premises* são essencialmente ambientes baseados na organização que fornecem recursos de computação, armazenamento e rede que podem ser adaptados às necessidades internas da organização.”

Ao escolher um *data center on-premises,* para atender a grandes picos de demanda, se faz necessário construir uma infraestrutura fixa que suporte tais picos, o que resultaria em uma subutilização em todo o resto do tempo em que a demanda

seria menor, além de que uma infraestrutura fixa limita o crescimento orgânico de picos de demanda, sendo necessário estudos recorrentes e expansões. Observa-se então que a elasticidade além de prover poder computacional, segurança e tranquilidade para a disponibilidade de aplicações, gera também economia, dado que o custo é realizado com base na quantidade de recursos que foram de fato consumidos por dado tempo.

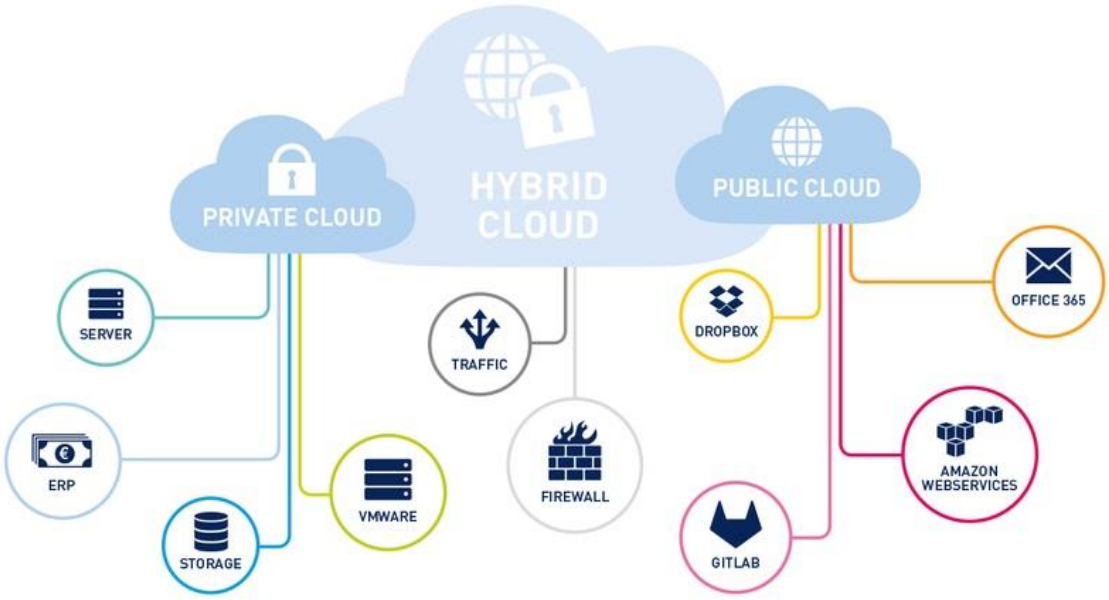
2.2.2 Cloud Computing

Segundo Rashid e Chaturvedi (2023, p. 421, tradução nossa), a *cloud computing,* do português computação em nuvem, é um termo que, de maneira simples, significa acessar, processar e armazenar dados em servidores distribuídos pela Internet, ao invés de realizar essas ações em servidores próprios localizados em instalação física própria.

Em *cloud computing*, abre-se várias possibilidades, pois esta facilmente pode atender as necessidades de utilização de clientes de pequeno porte até os de grande porte, e entre eles os clientes que têm suas demandas altamente oscilantes.

No mais, a *cloud* ainda atende aos mais diversos tipos de implantação de aplicações, inclusive sistemas e aplicações legadas. A figura 2.1 ilustra três tipos de *cloud computing* e exemplos de uso.

Figura 2.1 – Tipos de *Cloud Computing*



Fonte: 4Linux, 2023

2.2.2.1 Cloud Privada (Private Cloud)

Rashid e Chaturvedi (2023, p. 421, tradução nossa) define “*cloud* privada: este tipo de *cloud* trabalha para organização ou negócio definido, ou seja, *cloud* para organização específica”. A *cloud* privada nada mais é que uma infraestrutura computacional direcionada e especializada a atender apenas uma única empresa, divergindo do conceito de *data center on-premises* pelo fato de possuir local físico próprio e não estar localizado exatamente na mesma área em que se encontra a empresa em si.

2.2.2.2 Cloud Pública (Public Cloud)

Segundo Google (2023h, tradução nossa): “A nuvem pública é um tipo de computação em que os recursos são oferecidos por um provedor terceirizado pela Internet e compartilhados por organizações e indivíduos que querem usá-los ou comprá-los. Alguns recursos de *cloud computing* pública estão disponíveis gratuitamente, e os clientes podem pagar por outros recursos por meio de modelos de preços de assinatura ou pagamento por uso.”

Ou seja, diferentemente da *cloud* privada, a *cloud pública* fornece serviços a diferentes clientes por meio do compartilhamento de sua infraestrutura distribuída ao redor do mundo.

2.2.2.3 Cloud Híbrida (Hybrid Cloud)

A *cloud* híbrida é um modelo usualmente adotado por consumidores que não querem deixar de utilizar completamente sua *cloud* privada, muitas vezes deixando a parte de armazenamento de dados nesta e a parte de processamento para a *cloud* pública. Segundo Rinivasan, MD. e Varadarajan (2023, p. 44, tradução nossa).

> Um modelo de computação em nuvem híbrida lida com a hibridização ou mistura de nuvem privada e pública para um funcionamento mais proeminente e eficiente das nuvens. Em termos mais simples, o modelo híbrido é principalmente uma nuvem privada que permite que uma organização acesse uma nuvem pública como e quando necessário para compartilhamento de informações. Esse modelo fornece um meio mais eficiente de manter dados e aplicativos seguros.

2.2.2.4 Multicloud

A *multicloud* é uma resposta aos desafios encontrados na *cloud computing*, como a dependência de um único *cloud provier* e as limitações de desempenho, escalabilidade e segurança que podem surgir a partir dessa dependência.

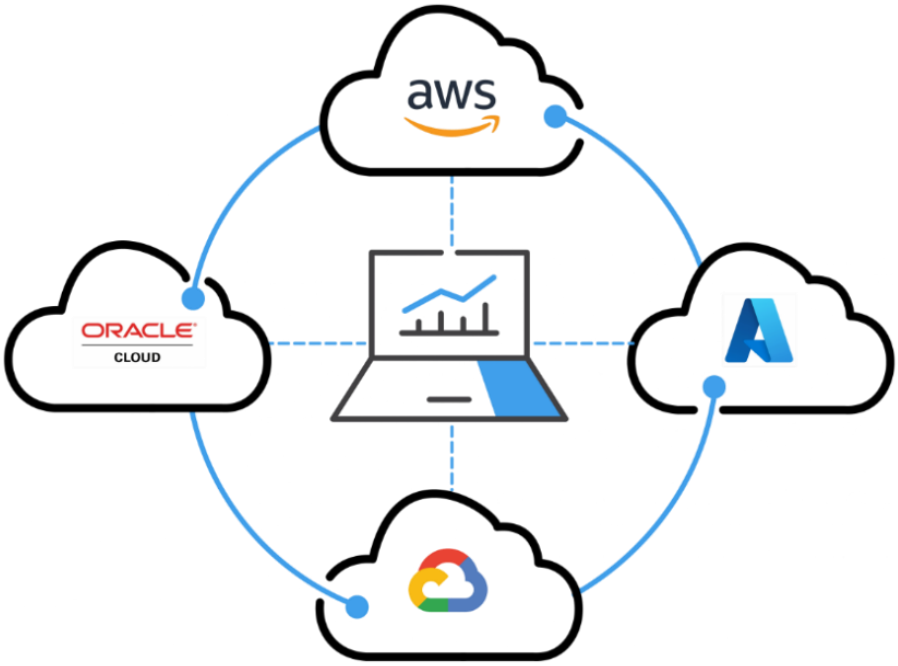
Ao adotar uma abordagem *multicloud*, uma organização pode explorar as vantagens oferecidas por diferentes provedores de nuvem, como preços competitivos, recursos especializados e geolocalização, para otimizar a infraestrutura de TI e atingir seus objetivos de negócios de forma com que não dependam exclusivamente de uma única *cloud provider.*

Oracle (2023) explica:

> A *multicloud* é uma estratégia de computação na nuvem que utiliza os melhores serviços de mais de um provedor de nuvem para implementar uma solução. Normalmente, a estratégia é orientada por requisitos de carga de trabalho, negócios e governança de dados. Uma solução multicloud integra IaaS, PaaS e SaaS em uma arquitetura bastante, ou pouca, acoplada. Uma solução *multicloud* bem projetada deve considerar a rede, o desempenho, a segurança, o gerenciamento operacional e o custo total de propriedade.

Ou seja, o objetivo da *multicloud* é extrair o melhor custo-benefício de cada *cloud provider* para as demandas de negócio. A figura 2.2 ilustra a *multicloud*.

Figura 2.2 – *Multicloud*



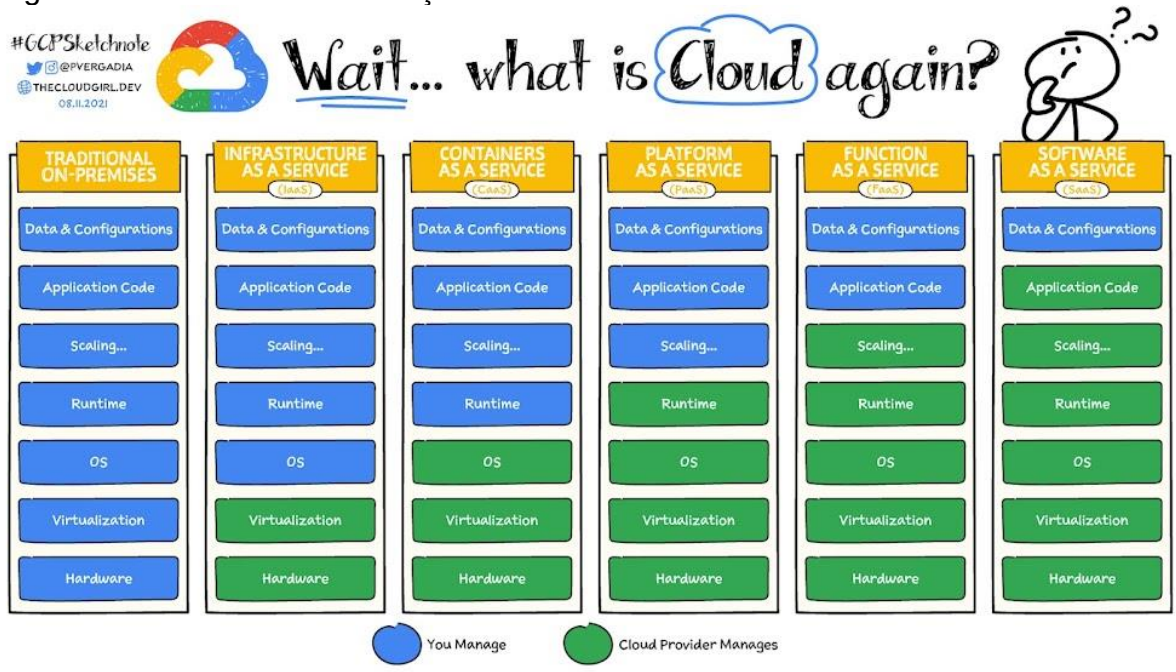
Fonte: Outcomex, 2023

2.2.2.5 Modelos de Serviços

Os modelos de serviços da *cloud computing* especificam funcionalidades e também quais partes do serviço são de responsabilidade do usuário e quais são de responsabilidade da *cloud provider,* como pode se observar na figura 2.3. Os usuários podem escolher os modelos que melhor atender às suas necessidades. Google (2023d, tradução nossa) específica:

> A *cloud computing* tem três modelos principais de serviços em *cloud*: IaaS *(Infrastructure as a Service*) PaaS (*Platform as a Service*) e SaaS (*Software as a Service*). Também podemos ouvir IaaS, PaaS e SaaS chamadas de ofertas de serviço em *cloud* ou categorias de *cloud computing*, mas todos esses termos se referem a como você usa a *cloud* na sua organização e o grau de gerenciamento é responsável nos seus ambientes de *cloud*.
> Além dessas três categorias amplas, você também pode encontrar outros tipos de serviços em *cloud* que incorporam outras tecnologias, como *containers*. Por exemplo, a crescente adoção de *containers* e arquiteturas de microsserviços levou ao surgimento do CaaS (*Container as a Service*).

Figura 2.3 – Modelos de serviços



Fonte: The Cloud Girl, 2023

2.2.2.5.1 IaaS

Infrastructure as a Service (IaaS), do português infraestrutura como serviço, é um modelo de *cloud computing* que fornece recursos de infraestrutura virtualizada sob

demanda pela Internet. A IaaS permite que os usuários provisionem e gerenciem suas próprias máquinas virtuais (VMs) e recursos de infraestrutura de forma flexível e personalizada, sem a necessidade de investir em *hardware* físico ou lidar com sua manutenção. Os usuários têm controle total sobre o sistema operacional e as configurações de *software* das VMs provisionadas.

Google (2023d, tradução nossa) define:

> A *Infrastructure as a Service*, ou IaaS, fornece recursos de infraestrutura sob demanda às organizações por meio da *cloud*, como computação, armazenamento, rede e virtualização. Os clientes não precisam gerenciar, manter ou atualizar a própria infraestrutura de *data center*, mas são responsáveis pelo sistema operacional, *middleware*, máquinas virtuais e quaisquer aplicativos ou dados.

2.2.2.5.2 PaaS

Platform as a Service (PaaS), do português plataforma como serviço, é um modelo de *cloud computing* que simplifica o processo de desenvolvimento, implantação e execução de aplicativos, fornecendo uma plataforma completa e gerenciada pelo *cloud provider*. Isso permite que os desenvolvedores se concentrem no desenvolvimento de aplicativos, enquanto a infraestrutura subjacente é tratada pelo provedor de PaaS.

Google (2023d, tradução nossa) define:

> A *Platform as a Service*, ou PaaS, fornece e gerencia todos os recursos de *hardware* e *software* para desenvolver aplicativos pela *cloud*. Os desenvolvedores e as equipes de operações de TI podem usar PaaS para desenvolver, executar e gerenciar aplicativos sem precisar criar e manter a infraestrutura ou a plataforma por conta própria. Os clientes ainda precisam escrever o código e gerenciar dados e aplicativos, mas o ambiente para criar e implantar apps é gerenciado e mantido pelo *cloud provider*.

2.2.2.5.3 SaaS

Software as a Service (SaaS) é um modelo de entrega de *software* baseado em *cloud* que oferece aos usuários acesso a aplicativos de *software* hospedados remotamente. Nesse modelo, em vez de instalar e manter o *software* em seus próprios dispositivos ou servidores, os usuários podem acessar os aplicativos pela Internet.

No SaaS, a *cloud provider* é responsável pela infraestrutura, manutenção e segurança do *software*, permitindo que os usuários se concentrem em usar o aplicativo sem se preocupar com problemas técnicos. Os aplicativos são hospedados em servidores de *cloud provider* e disponibilizados aos usuários por meio de um navegador da web ou de uma interface de usuário dedicada.

Os usuários do SaaS geralmente pagam uma taxa de assinatura para acessar o *software*, com base em um modelo de pagamento por uso ou em uma base de assinatura periódica. Isso proporciona aos usuários flexibilidade, já que podem aumentar ou diminuir o número de assinaturas de acordo com suas necessidades.

O SaaS é amplamente utilizado em várias áreas, como gerenciamento de relacionamento com o cliente (CRM), gestão de recursos humanos (RH), contabilidade, colaboração e produtividade. Esse modelo de entrega de *software* oferece vantagens como escalabilidade, atualizações automáticas, acesso universal e redução de custos, tornando-se uma opção popular para empresas e usuários finais.

Google (2023d, tradução nossa) define:

> O *Software as a Service*, ou SaaS, fornece toda a pilha de aplicativos, oferecendo um aplicativo baseado na *cloud* que os clientes podem acessar e usar. Os produtos SaaS são totalmente gerenciados pelo *cloud provider* e estão prontos para uso, incluindo todas as atualizações, correções de *bugs* e manutenção geral. A maioria dos aplicativos SaaS é acessada diretamente por meio de um navegador da Web, o que significa que os clientes não precisam fazer o *download* nem instalar nada nos dispositivos deles.

2.2.2.5.4 Outros Modelos

- *Container as a Service* (CaaS)

Modelo de serviço em *cloud computing* que simplifica a implantação, execução e gerenciamento de *containers* de aplicativos, proporcionando flexibilidade, escalabilidade e eficiência para os desenvolvedores.

Red Hat (2020) define: “*Container as a Service* (CaaS) é um serviço em *cloud* que ajuda a gerenciar e implantar aplicações usando abstração baseada em *container*. O CaaS pode ser implantado *on-premises* ou na *cloud*.”

- *Database as a Service* (DBaaS)

Modelo de serviço em *cloud computing* que simplifica o provisionamento e o gerenciamento de bancos de dados, permitindo que os usuários se concentrem em

suas aplicações e dados, enquanto a infraestrutura de banco de dados é cuidada pelo provedor de serviços em nuvem.

Nutanix (2023, tradução nossa) explica: “O termo “*Database as a Service*” (DBaaS) refere-se a *software* e/ou serviços que permitem que usuários configurem, operem e escalem bancos de dados sem precisar configurar *hardware* físico, instalar *softwares* ou configurar para desempenho”.

- *Function as a Service* (FaaS)

Modelo de *cloud computing* que oferece aos desenvolvedores a capacidade de executar funções individuais de código de maneira eficiente, escalável e sem a necessidade de gerenciar a infraestrutura subjacente. Ele oferece maior agilidade no desenvolvimento de aplicativos e reduz a sobrecarga operacional, permitindo que as equipes se concentrem no desenvolvimento de lógica de negócios e inovação.

IBM (2023) explica que: “FaaS, ou *Function as a Service*, é um serviço de *cloud computing* que permite aos clientes executar código em resposta a eventos, sem gerenciar a infraestrutura complexa normalmente associada à criação e lançamento de aplicativos de microsserviços”.

2.3 ARQUITETURAS DE DESENVOLVIMENTO

A arquitetura de desenvolvimento de *software* é a estrutura ou o plano geral que define a organização, o design, os componentes e as interações de um sistema de *software*.

Ela serve como um guia para o desenvolvimento, ajudando a garantir que o software atenda aos requisitos de negócio, seja escalável, seguro e eficiente. Além disso, a arquitetura de *software* também envolve a seleção e a definição das tecnologias, plataformas, padrões e protocolos utilizados no desenvolvimento.

Abordar questões como a organização dos dados, a estrutura do código, a separação de preocupações, a segurança, a escalabilidade, a tolerância a falhas, a eficiência, entre outros aspectos importantes para o sucesso do projeto.

Nas últimas décadas, o desenvolvimento de *software* tem passado por uma transformação significativa, com a transição de arquiteturas legadas para arquiteturas

cloud native. A arquitetura legada é baseada em tecnologias e práticas tradicionais, enquanto a *arquitetura cloud native* aproveita as vantagens da *cloud computing*.

2.3.1 Legada

Objective (2023) explica que um sistema legado é uma plataforma obsoleta, em uso por muitos anos e que devido aos vários avanços da tecnologia acaba deixando de se encaixar as necessidades da organização e/ou das aplicações que acabam utilizando-a.

A arquitetura legada é caracterizada por sistemas monolíticos, onde todas as funcionalidades estão contidas em um único componente ou aplicação, o código-fonte tende a ser altamente acoplado, dificultando a manutenção e a evolução do *software*. Além disso, a escalabilidade é limitada, exigindo a adição de recursos físicos para aumentar a capacidade do sistema.

Aplicações legadas dificilmente possuem suporte técnico especializado pela fabricante e, como são antigas, tendem a não ser escaláveis e nem otimizadas/compatíveis com as mais recentes tecnologias em execução no mercado. Deste modo, este tipo de aplicação perde o poder de integração e aumenta o grau de isolamento da aplicação, impactando diretamente no aproveitamento de novas tecnologias e recursos.

2.3.2 Cloud Native

A *cloud native* é uma abordagem em contrapartida aos sistemas legados. A arquitetura *cloud native* faz uso dos recursos disponibilizados pela *cloud computing*, como a escalabilidade, a elasticidade, a alta disponibilidade e os serviços gerenciados.

Os sistemas *cloud native* são implantados em ambientes de *cloud*, onde podem se beneficiar da orquestração de *containers*, permitindo uma implantação ágil e automatizada. Isso proporciona uma escalabilidade mais eficiente, onde os recursos podem ser dimensionados de acordo com a demanda, evitando gastos desnecessários.

A c*loud native* é um termo usado para descrever um *software* que é construído para ser executado em um ambiente de *cloud computing* e o qual são projetados

(*software)* para serem escaláveis, altamente disponíveis e fáceis de gerenciar (GITLAB, 2023). Portanto, há uma forma de desenvolvimento e adequação de aplicações para poderem extrair o máximo das qualidades e capacidades da *cloud computing*.

2.3.3 Legada x Cloud Native

As aplicações legadas remetem a utilização da abordagem arquitetural monolítica, sendo as aplicações consideradas monolitos. Já a abordagem *cloud native* foca na utilização de *container* e microsserviços. Para entender melhor a questão de conteinerização é necessário fazer uma comparação com virtualização. Ambas as tecnologias são amplamente utilizadas na área de computação e têm como objetivo otimizar o gerenciamento de recursos e a implantação de aplicações.

Tais abordagens têm transformado a maneira como as organizações desenvolvem e operam seus ambientes de computação, oferecendo vantagens significativas em termos de flexibilidade, eficiência e escalabilidade.

2.4 VIRTUALIZAÇÃO

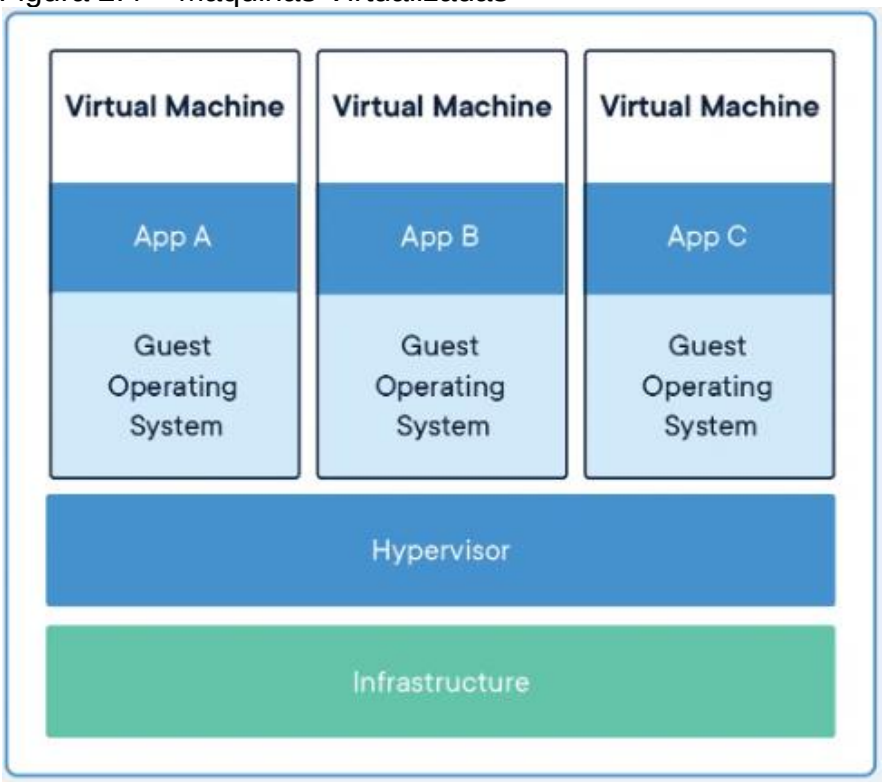
A virtualização é uma técnica que permite a criação de máquinas virtuais (VMs), proporcionando a capacidade de executar múltiplos sistemas operacionais em um único servidor físico. Essa abordagem é realizada através de um *software* chamado *hypervisor*, que atua como uma camada intermediária entre o *hardware* e as VMs.

Cada VM possui seu próprio sistema operacional e recursos alocados, permitindo a execução independente de aplicações e a maximização do uso de recursos de *hardware*. A figura 2.4 ilustra um exemplo com três máquinas virtuais, compostas pelas aplicações e sistemas operacionais, sob um *hypervisor* e sua infraestrutura.

Tanenbaum e Bos (2016, p. 325) explicam:

> A virtualização permite que um único computador seja o hospedeiro de múltiplas máquinas virtuais, cada uma executando potencialmente um sistema operacional completamente diferente. A vantagem dessa abordagem é que uma falha em uma máquina virtual não derruba nenhuma outra.

Figura 2.4 – Máquinas Virtualizadas



Fonte: Docker, 2023

2.5 CONTEINERIZAÇÃO

Por outro lado, a conteinerização é uma tecnologia que encapsula aplicações e suas dependências em unidades isoladas chamadas de *containers*. Esses *containers* compartilham o mesmo *kernel* do sistema operacional hospedeiro, mas são isolados uns dos outros, fornecendo uma camada de abstração e independência entre as aplicações.

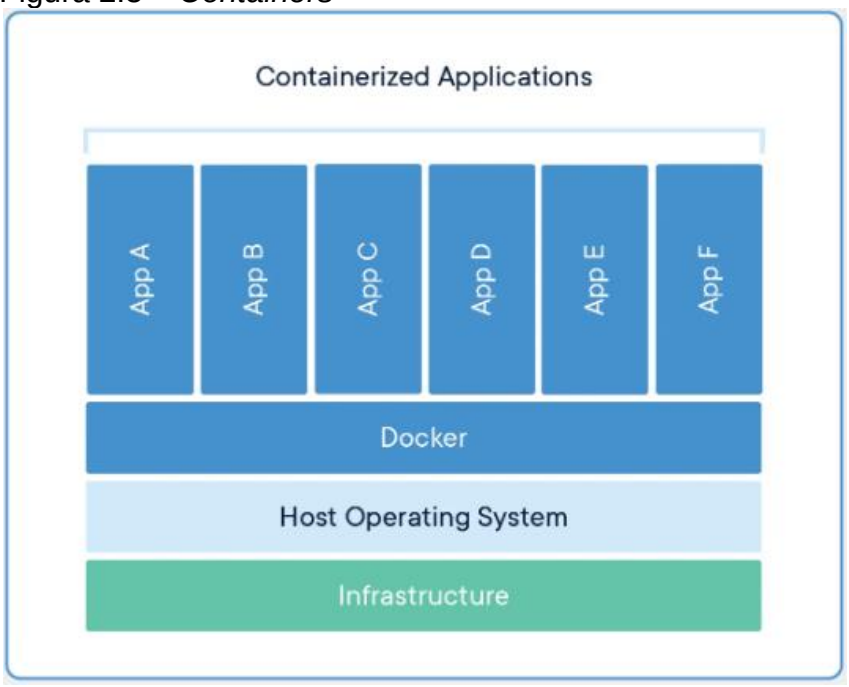
Docker (2023, tradução nossa), referência de mercado em *container*, explica:

> Um *container* é uma unidade padrão de *software* que agrupa o código e todas as suas dependências para que o aplicativo seja executado de forma rápida e confiável de um ambiente de computação para outro. Uma imagem de *container* do Docker é um pacote de *software* leve, autônomo e executável que inclui tudo o que é necessário para executar um aplicativo: código, tempo de execução, ferramentas do sistema, bibliotecas do sistema e configurações.

Com o Docker, é possível criar imagens de *containers* que são completamente reproduzíveis em qualquer ambiente. Isso significa que você pode garantir que o aplicativo funcionará exatamente da mesma maneira em ambiente de

desenvolvimento, homologação e produção, reduzindo problemas de incompatibilidade e facilitando a depuração. A figura 2.5 ilustra aplicações *(containers)* em um Docker.

Figura 2.5 – *Containers*



Fonte: Docker, 2023

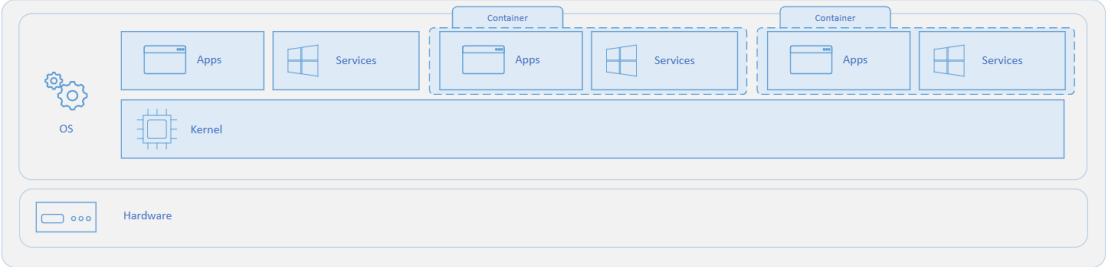
2.6 VIRTUALIZAÇÃO X CONTEINERIZAÇÃO

Tanto a conteinerização quanto a virtualização tem suas vantagens e desvantagens, além de casos de uso específicos.

Microsoft (2023a) explica que, diferentemente dos *containers*, a virtualização oferece isolamento completo entre as máquinas virtuais pois executa um sistema operacional com *kernel* próprio. Isto proporciona maior segurança e flexibilidade na execução de diferentes sistemas operacionais e aplicações, porém, implica em um consumo de recursos mais elevado devido à necessidade de emular um ambiente completo para cada máquina virtual.

Já a conteinerização é mais leve em termos de recursos, permitindo uma implantação rápida e escalonável de aplicações. No entanto, Microsoft (2023a) explica que os *containers* compartilham o mesmo *kernel* do sistema operacional do *host* (figura 2.6), o que pode resultar em um isolamento menos rigoroso entre as aplicações.

Figura 2.6 – *Containers* compartilhando *kernel*



Fonte: Microsoft, 2023

2.7 MICROSSERVIÇOS

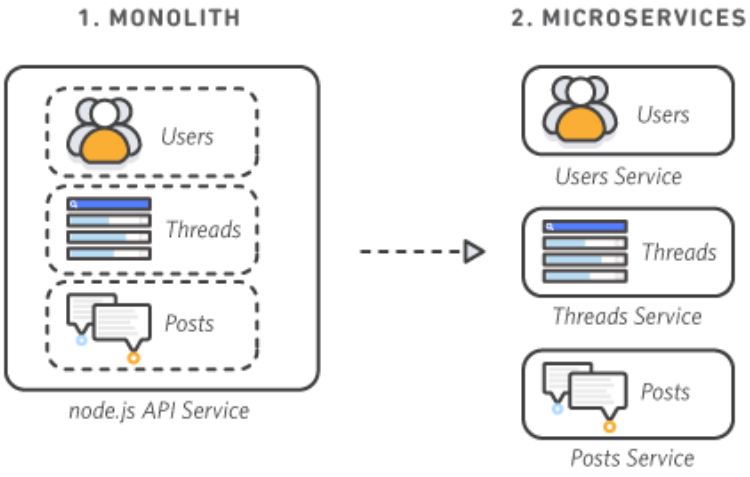
Uma aplicação é dividida em microsserviços que se comunicam, possibilitando o foco no desenvolvimento, implementação, manutenção e atualização de cada um desses microsserviços que a compõe, minimizando drasticamente o *downtime* pois raramente é necessário desabilitar todos os serviços da aplicação simultaneamente.

Red Hat (2023) explica:

> Microsserviços são uma abordagem de arquitetura para a criação de aplicações. Como um framework de arquitetura, microsserviços são distribuídos e levemente acoplados para que as mudanças feitas por uma equipe não corrompam toda a aplicação. A vantagem de usar microsserviços é que as equipes de desenvolvimento conseguem criar rapidamente novos componentes de app para atender às dinâmicas necessidades empresariais.

Na Figura 2.7 é ilustrado uma aplicação com arquitetura monolítica particionada em três microserviços.

Figura 2.7 – Monolito x Microsserviços



Fonte: Amazon Web Services (AWS), 2023

2.8 INFRAESTRUTURA COMO CÓDIGO (IaC)

Hashicorp (2023d, tradução nossa) explica que as ferramentas *Infrastructure as Code*, ou IaC, são ferramentas que permitem o gerenciamento de infraestrutura com arquivos de configuração em código substituindo as *Graphical User Interface* (GUI), do português interface gráfica de usuário. A ferramenta IaC possibilita que a infraestrutura seja criada, atualizada e gerenciada sendo consistente, reutilizável e segura no processo de provisionamento de infraestruturas.

Vale ressaltar que, cada *cloud provider* possui suas interfaces gráficas e seus *softwares* de linha de comando próprio para poder provisionar e interagir com seus recursos, entretanto, esta não é a maneira mais automatizada de gerenciar infraestruturas. Por meio das ferramentas IaC é possível criar códigos dos mais diversos componentes e recursos disponíveis nas *clouds*, de modo com que possam ser versionados, automatizados e reutilizados entre si e para outros projetos. No mais, as ferramentas IaC são tão versáteis e integradas que gerenciam ambientes *multicloud* (ambientes que são compostos de recursos provindos de duas ou mais *clouds* distintas).

Neste trabalho é provisionado uma infraestrutura na Google Cloud Platform (GCP), a *cloud provider* do Google, juntamente com a ferramenta IaC Terraform que é criada e mantida pela empresa HashiCorp. De maneira resumida, a função principal da ferramenta IaC é de realizar o provisionamento dos recursos através de comandos executados via linhas de código, que são passíveis de versionamento, sendo uma forma simplificada em relação ao tradicional acesso individual de diferentes plataformas e consoles.

3 TECNOLOGIAS

Esta seção tem como objetivo apresentar as principais tecnologias utilizadas no desenvolvimento deste projeto, sendo elas: Google Cloud Platform (GCP) e Terraform, Kubernetes e Zabbix.

3.1 GOOGLE CLOUD PLATFORM

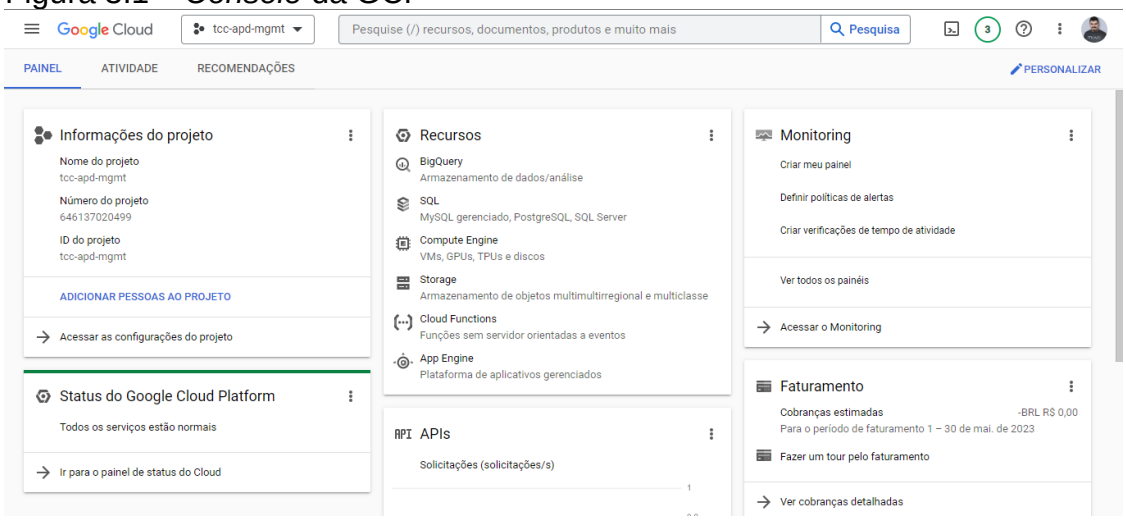
A Google Cloud Platform (GCP) é uma *cloud provider* que fornece uma variedade de serviços e recursos para ajudar indivíduos e organizações a construir, implantar e gerenciar aplicativos e serviços na nuvem.

Google (2023f) define: “O Google Cloud consiste em um conjunto de recursos físicos (computadores e unidades de disco rígido) e recursos virtuais, como máquinas virtuais (VMs), localizados nos data centers do Google por todo o mundo”.

Em termos científicos, a GCP pode ser descrita como uma infraestrutura de computação distribuída que utiliza uma rede de *data centers* interconectados para fornecer recursos computacionais escaláveis e flexíveis. Esses recursos incluem poder de processamento, armazenamento de dados, redes, serviços de aprendizado de máquina, análise de dados e outros serviços relacionados.

A figura 3.1 ilustra painel inicial do Google Cloud Platform do projeto tcc-apd-mgmt, onde há informações sobre o projeto e alguns atalhos para recursos do GCP.

Figura 3.1 – *Console* da GCP



Fonte: Elaborada pelo autor, 2023.

3.2 TERRAFORM

Hashicorp (2023c, tradução nossa) define: “Terraform é a ferramenta infraestrutura como código da HashiCorp. Ele permite que você defina recursos e infraestrutura em arquivos de configuração declarativos legíveis por humanos e gerencie o ciclo de vida de sua infraestrutura.”.

O Terraform é uma ferramenta *open-source* desenvolvida para auxiliar a automação e o provisionamento de infraestruturas de maneira declarativa. Para este fim, o Terraform permite que os usuários definam sua infraestrutura como código, ou seja, toda a infraestrutura é descrita em um arquivo de configuração no formato de linguagem específica, denominada Hashicorp Configuration Language (HCL), fornecendo uma abordagem mais ágil e reprodutível para o gerenciamento da infraestrutura.

Inicialmente, por meio da HCL, deve-se definir os recursos desejados, tais como máquinas virtuais, redes, bancos de dados e serviços em *cloud*. Em seguida, esse arquivo é analisado e cria-se um plano de execução, comparando o estado atual da infraestrutura com o estado desejado descrito no arquivo de configuração. Com base nessa comparação, o Terraform realiza as alterações necessárias para alcançar o estado desejado, provisionando, modificando ou destruindo recursos conforme necessário.

O Terraform possui parcerias com as maiores *cloud providers* do mercado, contendo vasta documentação da integração com cada uma delas. Inclusive, a documentação conta com as boas práticas de desenvolvimento de código para que este seja claro e padronizado. Além destas documentações, há também o desenvolvimento de módulos públicos para utilização, sendo estes mais avançados e complexos causando muitas vezes uma extensão de código desnecessária para casos de uso mais simples da ferramenta.

Com exceção do módulo de regras de *firewall*, todos os módulos utilizados neste projeto são de autoria própria.

O código de desenvolvimento do Terraform é baseado em diferentes “blocos”, os quais destacam-se: *terraform, provider, resource, output, data, variable, local* e *module* cada um possuindo sua função específica e contribuindo para o projeto como um todo.

3.2.1 Comandos

A seguir são explicados os códigos essencias utilizados no processo de desenvolvimento com a ferramenta Terraform:

- `terraform fmt`

Formata automaticamente os códigos dentro dos arquivos Terraform, baseado no padrão HCL.

- `terraform init`

Inicializa o diretório de trabalho, atualizando os *providers* e os módulos utilizados no projeto.

- `terraform validate`

Informa se há erros de sintaxe de código, versão, etc.

- `terraform plan`

Apenas informa o plano de execução do projeto, ou seja, informa o que vai ser criado, alterado ou destruído baseado na infraestrutura existente.

- `terraform apply`

Exibe o plano de execução do projeto e pede permissão realizar a implantação da infraestrutura e seus recursos.

- `terraform destroy`

Destrói toda a infraestrutura provisionada.

- `terraform output`

Exibe informações expostas dos recursos.

- `terraform show`

Exibe o estado atual do plano salvo, fornecendo informações sobre as estruturas implantadas.

- `terraform state`

Exibe de forma detalha os recursos implantados.

- `terraform version`

Exibe a versão instalada do Terraform.

- `terraform refresh`

Atualiza o estado do Terraform baseado nos recursos implantados.

- `terraform providers`

Fornece informação dos *providers*.

3.2.2 Blocos

A seguir são explicados os principais blocos utilizados no processo de desenvolvimento com a ferramenta Terraform:

- Bloco *terraform:*

Este bloco permite a especificação de diferentes opções e comportamentos relacionados ao processo de provisionamento e gerenciamento de infraestrutura. Por exemplo, este serve para definir propriedades como a versão do Terraform a ser usada e as *cloud providers* (como AWS, Azure, Google Cloud, etc.) das quais são utilizados recursos.

Figura 3.2 – Bloco *terraform* do projeto de gerência

```
##############################################################
# -> BLOCO DE CONFIGURAÇÃO TERRAFORM

terraform {

  # DEFINIÇÃO DE VERSÃO MÍNIMA REQUISITADA DO TERRAFORM
  required_version = ">= 1.4.6"

  # IMPORTAÇÃO DA CLOUD PROVIDER GCP
  required_providers {
    google = {
      source  = "hashicorp/google"
      version = ">= 4.64.0"
    }
  }
}
```

Fonte: Elaborada pelo autor, 2023

- Bloco *provider:*

Este bloco é responsável por configurar as *clouds providers*, referenciadas no bloco *terraform*, permitindo a especificação das credenciais de autenticação (chave de acesso), a região, as configurações de rede e outras propriedades necessárias para se conectar e interagir com o *cloud provider*. Ele define as informações necessárias para que o Terraform possa se comunicar e executar as operações de criação, modificação ou destruição dos recursos definidos no código do projeto Terraform.

Figura 3.3 – Bloco *provider* do projeto de gerência

```
################################################
# -> BLOCO DE CONFIGURAÇÃO DA GCP

provider "google" {
  credentials = file(local.projects.credentials)
  project     = local.projects.mgmt_id
  region      = local.projects.main_region
  zone        = local.projects.main_zone
}
```

Fonte: Elaborada pelo autor, 2023

- Bloco *resource:*

Este bloco define um recurso gerenciado pelo Terraform e representa um elemento de infraestrutura, como uma instância de servidor, uma máquina virtual, um *bucket* de armazenamento, entre outros. No bloco *resource*, é especificado o tipo do recurso, um nome único para o bloco e suas propriedades específicas. Essas propriedades incluem, por exemplo, informações como a região em que o recurso será criado, a versão da imagem, as configurações de rede, as políticas de acesso e outras opções relevantes. Ao criar um bloco resource, o Terraform é instruído a provisionar e gerenciar esse recurso em um *cloud provider* específico. O Terraform se responsabiliza a criar, modificar ou destruir o recurso de acordo com a configuração definida no bloco *resource*, garantindo que a infraestrutura declarada esteja sempre alinhada com o estado desejado.

Hashicorp (2023a) enfatiza que os blocos *resource* são fundamentais para a implementação de Infraestrutura como Código (IaC), permitindo que sejam definidos, versionados e gerenciados recursos de infraestrutura.

Figura 3.4 – Bloco *resource* do módulo GCE (máquina virtual)

```
#####################################################################
# -> CRIAÇÃO DA INSTÂNCIA DO GOOGLE COMPUTE ENGINE (GCE)

resource "google_compute_instance" "main" {
  name                      = var.name
  description               = var.description
  machine_type              = var.machine_type
  allow_stopping_for_update = var.allow_stopping_for_update
  zone                      = var.zone

  tags = var.tags

  boot_disk {
    initialize_params {
      image = var.boot_disk_image
    }
  }

  network_interface {
    subnetwork_project = var.network_interface_subnetwork_project
    subnetwork         = var.network_interface_subnetwork

    access_config {
      nat_ip = var.network_interface_public_static_ip
    }
  }

  metadata_startup_script = file(var.startup_script)
}
```

Fonte: Elaborada pelo autor, 2023

- Bloco *output:*

Este bloco é utilizado para definir valores de saída que são gerados durante a execução do código Terraform, permitindo a exposição de informações relevantes ou de resultados calculados pelo Terraform para uso externo salvando a saída do bloco no arquivo terraform.tfstate. O uso de blocos *output* é útil para expor informações como endereços IP, URLs, chaves de acesso, nomes de recursos criados, entre outros.

Figura 3.5 – Bloco *output* do submódulo “subnet” do módulo “network”

```
output "out_subnet_mgmt_1_region" {
  description = "Expõe apenas a região da subnetwork."
  value       = google_compute_subnetwork.subnet_vpc_mgmt.region
}
```

Fonte: Elaborada pelo autor, 2023

- Bloco *data:*

Muito similar ao bloco *output,* este bloco é usado para consultar e obter informações de uma fonte externa durante a execução do código Terraform. Ele permite a busca e acesso a informações de fontes externas, como detalhes de uma instância existente, informações de uma tabela de banco de dados, ou até mesmo dados obtidos por meio de uma API, sendo útil quando se precisa destas informações para tomar decisões ou definir configurações em sua infraestrutura. O bloco proporciona a integração de dados vindos de diferentes fontes e a utilização de forma dinâmica durante a execução do Terraform, garantindo que sua infraestrutura seja configurada corretamente com base nas informações obtidas.

Figura 3.6 – Bloco *data* do projeto de gerência

```
data "google_compute_network" "vpc_prd" {
  project = "tcc-apd-mgmt"
  name    = "vpc-prd"
}
```

Fonte: Elaborada pelo autor, 2023

- Bloco *variable:*

Este bloco é responsável pela definição de variáveis que são parametrizadas e passadas para módulos e recursos. As variáveis viabilizam configurações reutilizáveis e flexíveis, permitindo a personalização de valores específicos sem modificar diretamente o código Terraform.

Figura 3.7 – Bloco *variable* do submódulo “instance” do módulo “cloud_sql”

```
variable "region" {
  description = "Região da instância do banco de dados."
  type        = string
}
```

Fonte: Elaborada pelo autor, 2023

- Bloco *local:*

É utilizado para a definição de variáveis locais que são usadas dentro do mesmo módulo ou arquivo de configuração. Além disso, este bloco permite a definição de valores intermediários ou computados que são usados em várias partes do código. O bloco local pode conter interpolação em sua definição.

Figura 3.8 – Bloco *local* do projeto de gerência

```
#####################################################################
# -> LOCALS UTILIZADAS NO PROJETO

locals {

  project = {
    id          = "tcc-apd-prd"
    credentials = "../../files/sak/sak-tcc-apd-prd-terraform.json"
    main_region = "us-central1"
    main_zone   = "us-central1-c"
  }
}
```

Fonte: Elaborada pelo autor, 2023

- Bloco *module*:

O módulo "block" é um recurso que permite agrupar e reutilizar configurações de recursos relacionados em uma unidade lógica, fornecendo uma maneira flexível e eficiente de criar e gerenciar recursos complexos em infraestruturas provisionadas pelo Terraform.

Este bloco pode ser invocado em várias partes do código do Terraform, fornecendo os valores adequados para os parâmetros configuráveis, se houver. Isso torna mais fácil e eficiente a criação e gerenciamento de recursos complexos, reduzindo a duplicação de código e facilitando a manutenção.

Este bloco pode ter como fonte módulos de autoria própria quanto módulos de terceiros encontrados na Internet.

Figura 3.9 – Bloco *module* “web_server” do projeto de gerência

```
# ~ CONFIGURAÇÃO DE WEB SERVER
module "web_server" {
  source = "../../modules/gce"

  name                                = "web-server"
  description                         = "Web Server p/ aplicações dinâmicas."
  machine_type                        = "e2-medium"
  allow_stopping_for_update           = true
  zone                                = local.project.main_zone
  tags                                = ["web-server", "ssh-open", "zabbix-client"]
  boot_disk_image                     = "debian-cloud/debian-11"
  network_interface_subnetwork_project = data.google_compute_subnetwork.subnet_prd_1.project
  network_interface_subnetwork        = data.google_compute_subnetwork.subnet_prd_1.name
  network_interface_public_static_ip  = module.web_server_spuia.out_address
  startup_script                      = "../../files/startup_script/web_server.sh"
}
```

Fonte: Elaborada pelo autor, 2023

3.2.3 Terraform State

O Terraform state é um componente fundamental da ferramenta Terraform. Hashicorp (2023b, tradução nossa) explica:

> O Terraform deve armazenar o estado sobre sua infraestrutura e configuração gerenciadas. Esse estado é usado pelo Terraform para mapear recursos do mundo real para sua configuração, acompanhar os metadados e melhorar o desempenho de grandes infraestruturas.

Ele é responsável por armazenar e rastrear o estado atual da infraestrutura provisionada pelo Terraform, contendo informações sobre os recursos que foram criados, suas configurações, dependências e metadados. É utilizado o estado para planejar e gerenciar alterações na infraestrutura. Ao executar um plano de implantação ou aplicar uma configuração, o Terraform compara o estado atual com a definição de infraestrutura declarada no código e determina quais ações precisam ser executadas para alcançar o estado desejado.

O Terraform state é armazenado em um arquivo chamado terraform.tfstate. Além disso, há também o terraform.tfstate.backup que armazena o estado do provisionamento anterior ao atual.

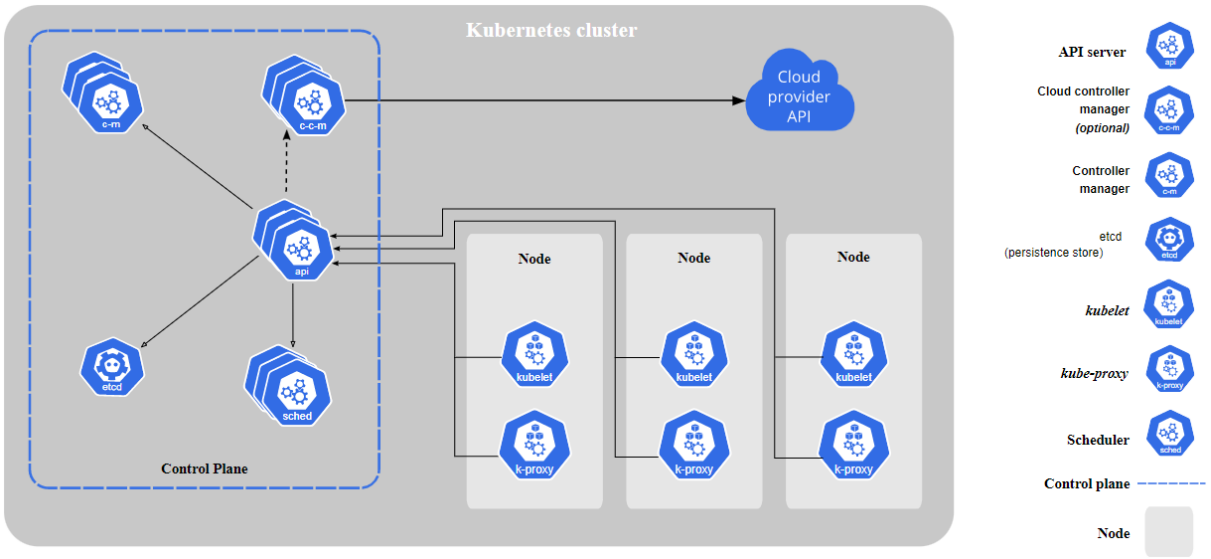
3.3 KUBERNETES

Kubernetes (2023b) diz que Kubernetes, ou K8s, é um sistema *open-source* utilizado para executar ações com *containers*, sendo uma ferramenta poderosa que ajuda a simplificar a implantação e o gerenciamento de aplicativos em *containers dentro de* um ou mais *nodes*.

Quando se cita microsserviços, refere-se a uma aplicação que possui vários *containers*, cada um contendo uma parte da aplicação e podendo estar em *nodes* diferentes. O Kubernetes atua como um maestro, organizando esses *containers,* nomeados de *pods* no Kubernetes, e garantindo que eles estejam sempre funcionando corretamente. Ele cuida automaticamente de coisas como balanceamento de carga, escalabilidade e recuperação de falhas, para que você possa se concentrar no desenvolvimento do seu aplicativo sem se preocupar com a infraestrutura subjacente.

Em essência, o Kubernetes ajuda a orquestrar todo o processo de execução de seus *containers* de maneira eficiente e confiável.

Figura 3.10 – Componentes do *cluster* com Kubernetes



Fonte: Kubernetes, 2023

3.3.1 Kubectl

Segundo Kubernetes (2023a), Kubectl é uma Command-Line Interface (CLI) fornecida pela propria Kubernetes para comunicação com o Kubernetes, de forma mais específica, com o control plane que nada mais é que o componente orquestrador do Kubernetes. Neste projeto, ele é a base para comunicação com o *cluster* Kubernetes, visto que o Helm, gerenciador de pacotes do Kubernetes, o utilizará para realizar a implantação das aplicações Ingress-nginx, Cert-manager e Rancher.

Figura 3.11 – Comando Kubectl para verificar consumo dos *nodes*

```
Kubectl: gke-prd-std-apps
> kubectl top nodes
W0602 04:21:45.961904      58 top_node.go:119] Using json format to get metrics. Next release will switch to protocol-buffers, switch early by passing --use-protocol-buffers flag
NAME                                                CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
gke-gke-prd-std-apps-gke-prd-node-poo-86479d21-3kbk   95m          4%     641Mi           10%
gke-gke-prd-std-apps-gke-prd-node-poo-86479d21-jk69   100m         5%     620Mi           10%
gke-gke-prd-std-apps-gke-prd-node-poo-a4907fbf-5v94   263m         13%    1037Mi          17%
gke-gke-prd-std-apps-gke-prd-node-poo-a4907fbf-pbqb   151m         7%     666Mi           11%
gke-gke-prd-std-apps-gke-prd-node-poo-fc055b96-vq17   211m         10%    866Mi           14%
gke-gke-prd-std-apps-gke-prd-node-poo-fc055b96-ws4h   123m         6%     687Mi           11%
>
```

Fonte: Elaborada pelo autor, 2023

3.3.2 Helm

Helm (2023) explica que Helm é um gerenciador de pacotes para Kubernetes, sendo estes pacotes apelidados de *charts*. O *chart* nada mais é que um conjunto de

arquivos que descrevem uma aplicação, sendo esta constituída de vários recursos do Kubernetes. Pode-se, por exemplo, implantar por meio do Helm aplicações como bancos de dados, APIs, *websites*, etc.

Neste projeto, o Helm será utilizado para instalar 3 aplicações no *cluster* de gerência, sendo estas:

- Ingress-nginx, aplicação que utiliza o Nginx como *proxy* reverso e *load balancer*;

- Cert-manager, aplicação que controla certificados SSL/TLS para Kubernetes;

- Rancher, plataforma para Kubernetes

Figura 3.12 – Comando Helm para instalação do Rancher.

```
helm upgrade --install rancher rancher-stable/rancher \
    --namespace cattle-system \
    --create-namespace \
    --version 2.7.3 \
    --set hostname=rancher.andreparradourado.com.br \
    --set bootstrapPassword=admin \
    --set ingress.ingressClassName=nginx \
    --set ingress.tls.source=letsEncrypt \
    --set letsEncrypt.email=andre.dourado@movti.com.br \
    --set letsEncrypt.ingress.class=nginx
```

Fonte: Elaborado pela autor, 2023.

3.3.3 Rancher

Rancher (2023) explica que o Rancher é uma *stack* completa para equipes gerenciarem múltiplos *clusters* Kubernetes, em ambientes diferentes. O Rancher possui uma interface gráfica amigável e que integra com Kubectl e o Helm, se tornando uma ferramenta completa em que se pode realizar implantação de aplicações, implantações de *charts* do Helm, monitorar *workloads* e monitorar o *cluster* e seus *nodes*. O Rancher foi instalado no *cluster* Kubernetes de gerência, com o objetivo de importar e monitorar o *cluster* Kubernetes de produção.

3.4 ZABBIX

Quando se trata de monitoramento e observabilidade são utilizadas aplicações que monitoram os recursos de hosts e URLs, visando observar a saúde dos mesmos

de forma com que se possa atuar de forma preventiva, preditiva e corretiva em relação a instabilidades e problemas. O Zabbix é a ferramenta escolhida para este projeto.

> O Zabbix é um software que monitora vários parâmetros de uma rede e a saúde e integridade de servidores, máquinas virtuais, aplicativos, serviços, bancos de dados, sites, *cloud* e muito mais. O Zabbix usa um mecanismo de notificação flexível que permite aos usuários configurar alertas baseados em e-mail para praticamente qualquer evento. Isso permite uma reação rápida aos problemas do servidor. O Zabbix oferece excelentes recursos de relatórios e visualização de dados com base nos dados armazenados. Isso torna o Zabbix ideal para planejamento de capacidade. (ZABBIX, 2023, tradução nossa)

4 IMPLEMENTAÇÃO

A fase de implementação foi executada em etapas. Realizou-se na primeira etapa as configurações iniciais que englobaram as configurações essenciais da *cloud provider* GCP, a configuração da máquina utilizada para desenvolvimento dos códigos e, por fim, o desenvolvimento do Shell Script que quando executado ativa todas as APIs utilizadas em cada um dos ambientes/projetos, nivelando-os.

Já na segunda etapa, foram desenvolvidos os módulos Terraform (e seus submódulos), visando as boas práticas que torna os módulos reutilizáveis entre diferentes projetos.

Na terceira etapa desenvolveu-se o ambiente de gerência, sendo este o mais complexo do projeto visto que hospeda toda a infraestrutura comum (VPCs, sub-redes, DNS, etc) utilizada. Além disso, no ambiente de gerência também se encontram os recursos que monitoram o ambiente de produção. Por fim, foi desenvolvido o ambiente de produção onde são hospedados: servidor Web, *cluster autopilot* Kubernetes, *cluster standard* Kubernetes, *cluster* Dataproc (ecossistema Hadoop) e Website estático com *bucket* e *load balancer* HTTPS.

4.1 CONFIGURAÇÕES INICIAIS

1) Preparação da Google Cloud Platform
   a. Criação da pasta mãe e dos projetos filhos (produção e gerência)
   b. Criação das contas de serviços utilizadas pelo Terraform para interação com GCP (produção e gerência) e geradas suas chaves JSON
   c. Ativação da API de secrets e armazenadas as chaves geradas para as contas de serviço
   d. Configuração das permissões iniciais para as contas do projeto

2) Preparação do ambiente de trabalho
   a. Instalação do Windows Subsystem for Linux 2 (WSL 2) no sistema operacional Windows 11
   b. Instalação do sistema operacional Ubuntu via WSL 2
   c. Instalação do *software* gcloud, CLI do Google Cloud Platform

d. Instalação do *software* Terraform
e. Instalação do *software* Kubectl (embutido na instalação da CLI do GCP)
f. Instalação do *software* Helm
g. Instalação do *software* Git
h. Criação do diretório de trabalho onde são armazenados todos os arquivos utilizados no projeto

3) Nivelação de projetos
   a. Configuração do *software* gcloud (vinculação com conta GCP)
   b. Desenvolvimento de *script* de equalização, que ativa APIs utilizadas em todos os projetos

4.2 DESENVOLVIMENTO DE MÓDULOS TERRAFORM

1) Network (Módulo para demandas de rede)
   a. Shared VPC (Define VPC compartilhada)
   b. VPC (Cria VPCs)
   c. Subnet (Cria subredes)
   d. Private Services Access (Estabelece conexão interna entre recursos do projeto e recursos de projetos nativos do Google)
   e. Static Public IP Address (Reserva endereços IP públicos para serem utilizados em recursos provisionados)
   f. DNS (Submódulo para demandas de DNS)
      i. DNS Público (Controlador de DNS público)
      ii. DNS Privado (Controlador de DNS privado)
      iii. Registros (Cria registros em zonas DNS)

2) IAM (Módulo para demandas de acesso e permissionamento)

3) GCS (Módulo para demandas de armazenamentos em buckets)
   a. Bucket (Cria *buckets)*
   b. Object (Faz o upload de arquivos para os *buckets*)

4) GCE (Módulo para demandas de máquinas virtuais)

5) Cloud SQL (Módulo para demandas de DBaaS)
    a. Database (Cria banco de dados dentro de SGBDs)
    b. Instance (Cria instâncias de SGBD)
    c. User (Cria usuários dentro de SGBDs)

6) GKE (Módulo para demandas de *cluster* Kubernetes)
    a. Autopilot (Cria *cluster* Kubernetes autogerenciado)
    b. Standard (Cria *cluster* Kubernetes standard)
        i. Node Pool (Cria *node pool* para *cluster* padrão)

7) Static Website (Módulo que provisiona *website* estático por meio de *bucket* e *load balancer*)

8) Dataproc (Módulo para demandas de *cluster* com ecossistemas Hadoop)

4.3 DESENVOLVIMENTO DE AMBIENTE DE GERÊNCIA

Nesta seção são abordados os tópicos levados em consideração para o desenvolvido do código Terraform do projeto de gerência.

4.3.1 Configuração de Shared VPC

Segundo Google (2023e, tradução nossa): “A Shared VPC permite que uma organização conecte recursos de vários projetos a uma *virtual private cloud* (VPC) comum para que eles possam se comunicar de maneira segura e eficiente usando endereços IP internos dessa rede.”

Para isto, é necessário definir um *host project* para compartilharmento e então anexar *service projects* a este *host project* para consumir os recursos compartilhados. Neste projeto o *host project* é o projeto de gerência (tcc-apd-mgmt), enquanto o projeto de produção (tcc-apd-prd) é um *service project.*

A utilização da Shared VPC é importante pois permite que as organizações apliquem e imponham políticas de controle de acesso consistentes no nível da rede para vários projetos de serviço na organização enquanto delegam responsabilidades administrativas.

4.3.2 Configuração de Virtual Private Clouds (VPCs)

Segundo Google (2023g, tradução nossa): “Uma rede Virtual Private Cloud (VPC) é uma versão virtual de uma rede física que é implementada dentro da rede de produção do Google usando Andromeda”. Neste projeto foi criado uma VPC para o ambiente de gerência e outra para o ambiente de produção.

4.3.3 Configuração de sub-redes

No ambiente de gerência foram configuradas duas sub-redes, sendo a primeira para recursos diversos, subnet-mgmt-1, e a segunda dedicada para o *cluster standard* Kubernetes de monitoramento, subnet-mgmt-k8s.

Para o ambiente de produção foram configuradas três sub-redes, a primeira para recursos diversos, subnet-prd-1, a segunda para o *cluster standard* Kubernetes, subnet-prd-k8s, e a terceira para o *cluster autopilot* Kubernetes, subnet-prd-k8s-ap.

4.3.4 Configuração de regras de Firewall

Foram configuradas regras de *firewall* que atendem aos critérios de segurança para cada aplicação existente no projeto. As regras de *firewall* englobam regras de *ingress*(entrada) e *egress*(saída), controlando as comunicações dos ambientes e recursos dos projetos.

Para isso foram avaliadas quais portas seriam necessárias para cada um dos casos levando em consideração os recursos que seriam provisionados.

4.3.5 Configuração de buckets para armazenamento de blobs

Buckets são os recipientes básicos que armazenam seus dados (GOOGLE, 2023a). Complementando, nos *buckets* pode-se armazenar arquivos blob para acesso e consumo, sendo estes de acesso privado ou público.

Em todos os casos do projeto de gerência foi adotada a política privada, onde implantou-se um *bucket* para armazenar logs e outro para armazenar *scripts*.

4.3.6 Configuração de DNS público e DNS privado

Configurou-se um DNS público, que utiliza um domínio público registrado, e um DNS privado que o domínio é reconhecido apenas internamente e nas VPCs definidas, sendo elas a VPC de gerência, vpc-mgmt, e a VPC de produção, vpc-prd.

4.3.7 Configuração de registros MX, SPF, DKIM para GWS

Google Workspace (GWS) é ambiente de trabalho que o Google fornece. Neste ambiente de trabalho há o Gmail, ferramenta de *e-mail*. Para que o disparo de *e-mails* ocorra adequadamente, é necessário inserir os registros Mail Exchange (MX), Sender Policy Framework (SPF) e Domain-based Message Authentication Reporting and Conformance (DKIM) no controlador DNS público do domínio.

4.3.8 Configuração de Private Service Access

O Private Service Acess serve para a infraestrutura do projeto comunicar internamente, via endereços IP privados, com serviços hospedados em redes do Google, sem ter que trafegar de forma externa pela Internet.

4.3.9 Reserva de endereços IP públicos e estáticos

Realizada para que as aplicações da infraestrutura tenham um endereço público acessável vinculada ao domínio público registrado.

4.3.10 Configuração de acesso e permissionamento via IAM

Concedeu-se acesso e permissionamento para as contas e contas de serviço com base no princípio do privilégio mínimo. Segundo Google (2023b), o princípio de privilégio mínimo é uma diretriz de segurança para a concessão de acesso onde deve-se atribuir apenas o acesso mínino necessário para que um usuário e/ou conta de serviço execute sua função.

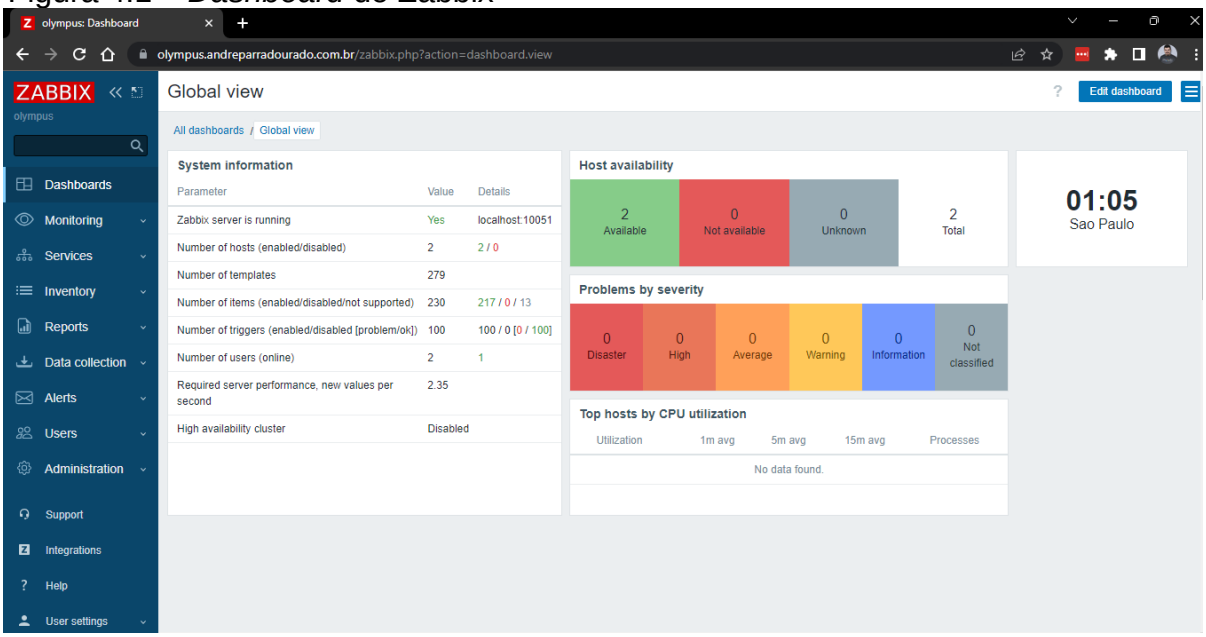
Em muitos casos, principalmente após a criação dos ambientes, essas permissões podem até ser revogadas pois o ambiente já foi construído e nada mais deve ser criado sem os devidos processos para isso.

4.3.11 Servidor Zabbix

Para o provisionamento do servidor Zabbix foi instanciado uma VM no Google Compute Engine (GCE), solução IaaS do Google. Nesta máquina virtual com sistema operacional Debian foram instalados e configurados, via *script* de inicialização, os serviços Nginx, *web server* para hospedagem do *website* do Zabbix, o Certbot, para administração de certificado SSL/TLS da máquina virtual, e a aplicação Zabbix. Em relação ao banco de dados da aplicação, se encontra em um DBaaS do Google, o Cloud SQL.

A figura 4.1 apresenta o *dashboard* informativo do Zabbix.

Figura 4.1 – *Dashboard* do Zabbix

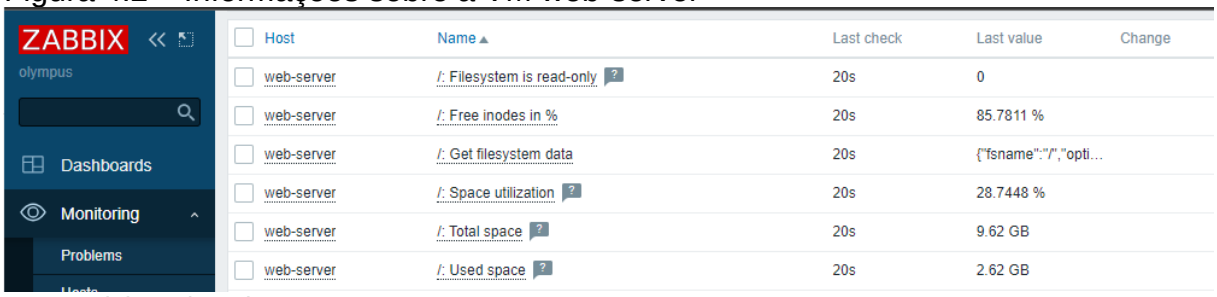


Fonte: Elaborada pelo autor, 2023

A interface inicial do Zabbix apresenta uma visão geral do monitoramento, contendo informações como *status* do próprio Zabbix, de *alertas* e *hosts*. Nesta tela é possível verificar a quantidade de alertas separados por sua criticidade. Além disso, é possível ver se há indisponibilidade afetando servidores.

Na figura 4.2 são apresentados os dados recentes referentes a máquina virtual web-server.

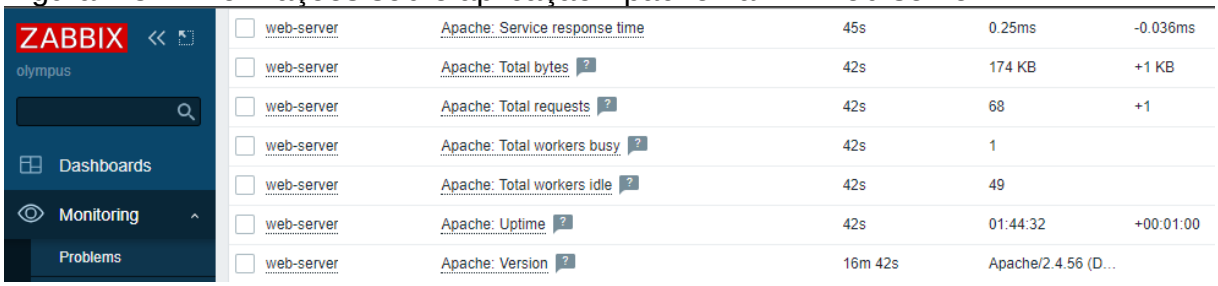
Figura 4.2 – Informações sobre a VM web-server



Fonte: Elaborada pelo autor, 2023

A figura 4.3 demonstra os dados recentes referentes a aplicação Apache encontrada na VM web-server.

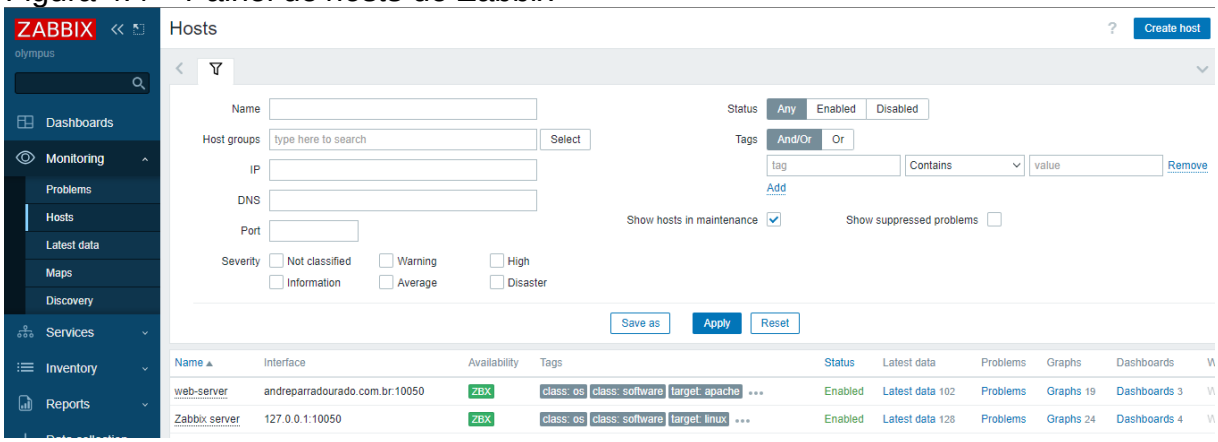
Figura 4.3 – Informações sobre aplicação Apache na VM web-server



Fonte: Elaborada pelo autor, 2023

A figura 4.4 demonstra o painel de *hosts* onde se encontra a lista completa de *hosts* monitorados pelo Zabbix e seus *status.*

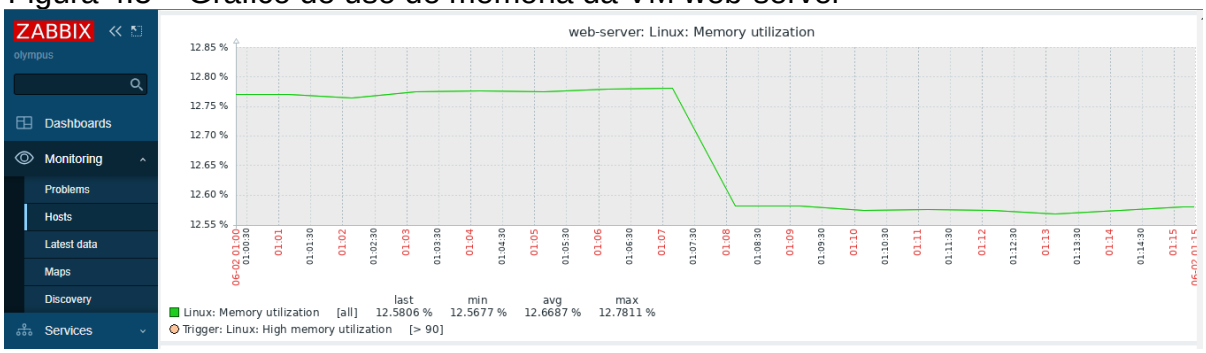
Figura 4.4 – Painel de *hosts* do Zabbix



Fonte: Elaborada pelo autor, 2023

No Zabbix há também gráficos que representam o comportamento de recursos, conforme mostrado na figura 4.5.

Figura 4.5 – Gráfico de uso de memória da VM web-server

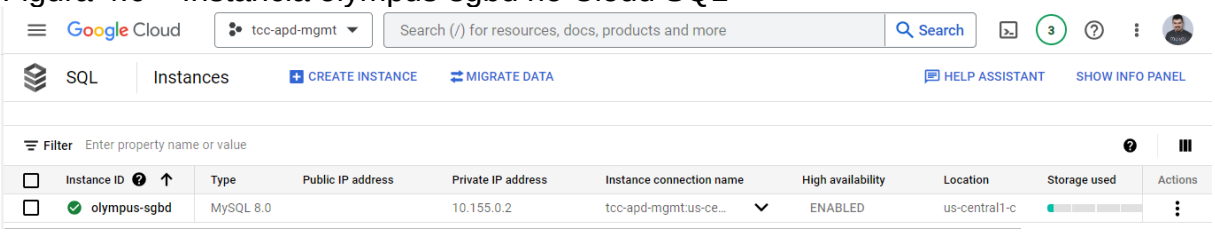


Fonte: Elaborada pelo autor, 2023

4.3.12 Banco de dados para servidor Zabbix

A aplicação Zabbix necessita de um banco de dados relacional para armazenar seus dados e os dados dos recursos que monitoram. Para isso, por meio Cloud SQL, solução DBaaS do Google, foi criado uma instância de banco de dados relacional para servir o servidor Zabbix (figura 4.6).

Figura 4.6 – Instância olympus-sgbd no Cloud SQL



Fonte: Elaborada pelo autor, 2023

4.3.13 Cluster standard Kubernetes com aplicação Rancher

Por meio do Google Kubernetes Engine (GKE), foi provisionado um *cluster standard* Kubernetes e associadamente um *node pool* preemptivo, um conjunto de *nodes*.

No cluster standard, o gerenciamento dos *worker nodes* é feito pelo próprio consumidor, porém o *control plane* é administrado pelo Google.

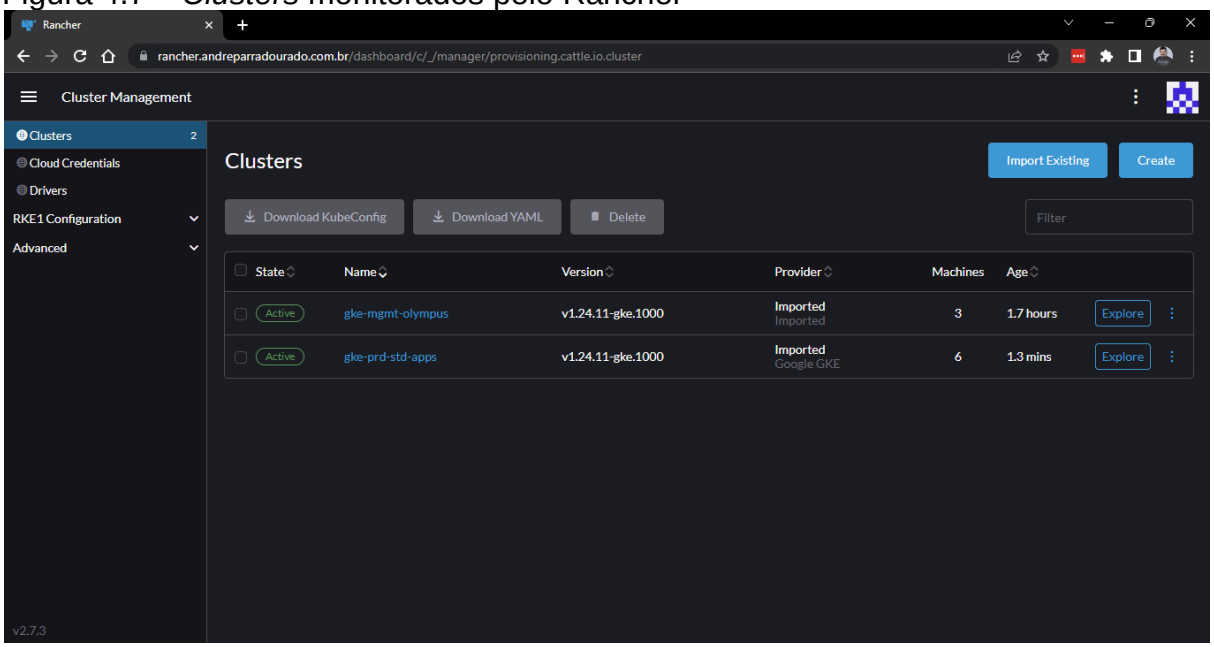
Antes da instalação do Rancher propriamente dito, foram instalados também os serviços pré-requisito do mesmo, Ingress-nginx e do Cert-manager, tudo por meio do gerenciador de pacotes Helm.

A figura 4.7 contém a tela do Rancher com a lista de *clusters* que estão sob monitoramento do mesmo, no caso sendo o próprio *cluster* de gerência e também o *cluster standard* de produção.

Por meio desta tela é realizado a importação de *clusters* para monitoramento, os *clusters* podem estar localizados em qualquer lugar, *data centers on-premises*, *clouds providers* distintas etc, tendo como pré-requisitos: comunicação possível e conta de serviço com permissão para acesso ao mesmo.

O Rancher é utilizado para os *clusters* de todos os ambientes, pois é ele que viabiliza o monitoramento das aplicações antes de serem implantadas no ambiente produtivo tanto quanto após a implantação.

Figura 4.7 – *Clusters* monitorados pelo Rancher

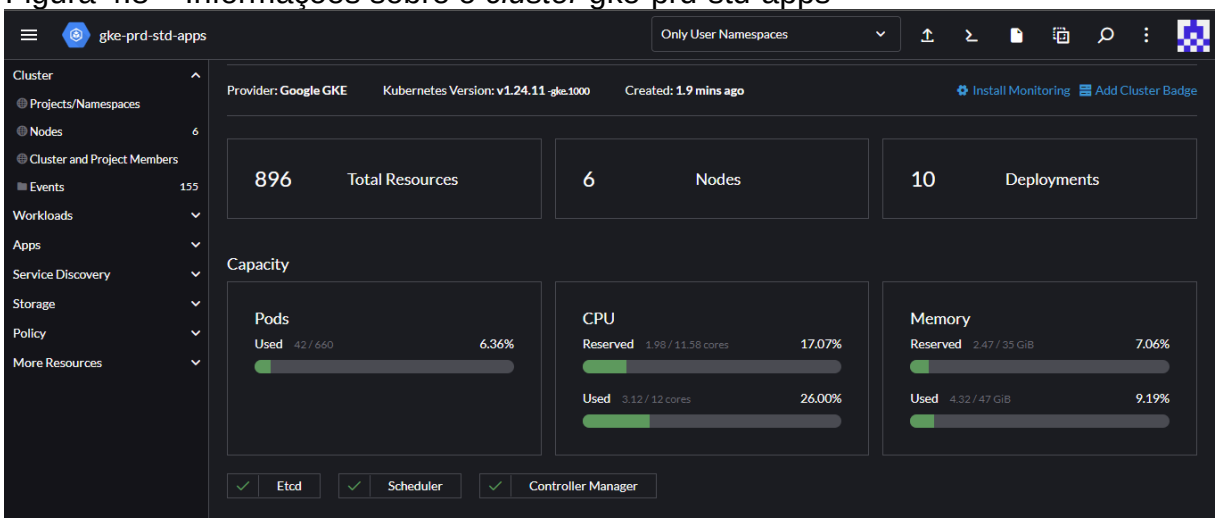


Fonte: Elaborada pelo autor, 2023

Ao acessar a página específica de um *cluster* são exibidas informações detalhadas do *cluster* tais como:

- Quantidade de *pods*
- Reserva/consumo de CPU e memória
- Total de recursos, *nodes* e *deployments*

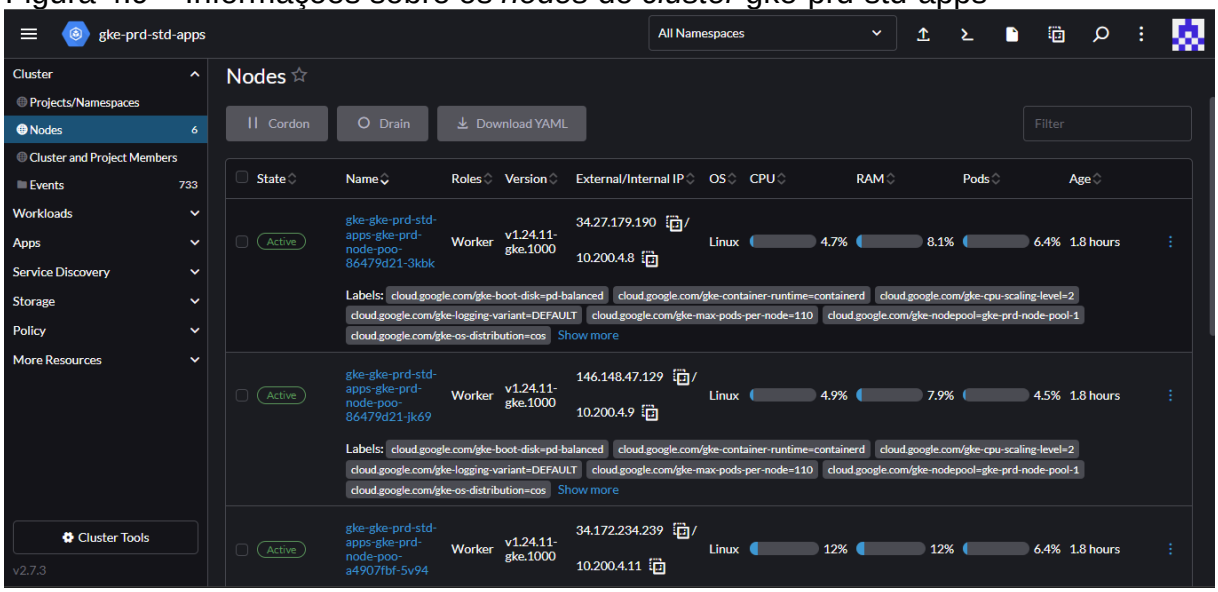
Figura 4.8 – Informações sobre o *cluster* gke-prd-std-apps



Fonte: Elaborada pelo autor, 2023

A figura 4.9 demonstra a lista de *nodes* do *cluster* de produção. Nesta tela há informações como *status* do *node*, nome, papel, versão, endereços IP, sistema operacional, consumo de CPU e de memória RAM, quantidade de *pods* e também a quantas horas ele está *online*.

Figura 4.9 – Informações sobre os *nodes* do *cluster* gke-prd-std-apps



Fonte: Elaborada pelo autor, 2023

4.3.14 Certificados SSL/TLS

Os certificados SSL são responsáveis pela comunicação “segura” (encryptada). O certificado SSL foi instalado no servidor Zabbix por meio do *software* Certbot

(software open-source que faz gestão e implantação de certificados da Let's Encrypt), e no *cluster* de gerência por meio do Cert-manager, uma aplicação que executa a mesma função do Certbot, porém a nível de *cluster* (utiliza também o Let's Encrypt).

4.4 DESENVOLVIMENTO DE AMBIENTE DE PRODUÇÃO

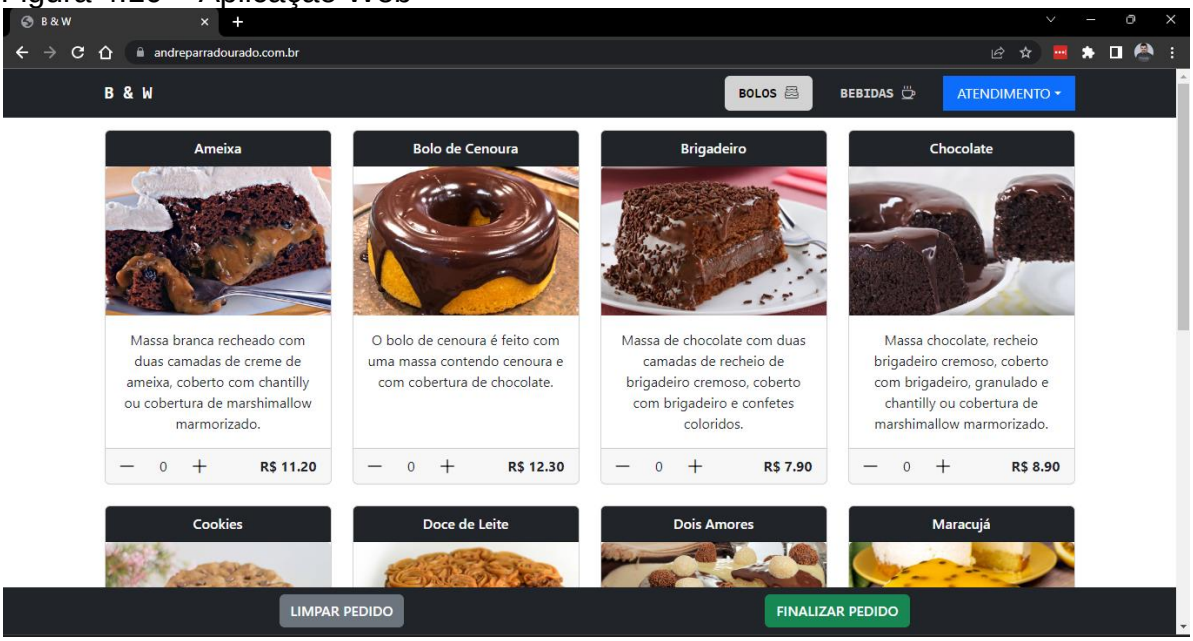
4.4.1 Servidor Web

Para o provisionamento do servidor Web foi instanciado uma VM no GCE. Nesta máquina virtual com sistema operacional Debian foram instalados e configurados, via *script* de inicialização, os serviços Apache2, *web server* para hospedagem do *website*, o Certbot, para administração de certificado SSL/TLS da máquina virtual, e a aplicação Zabbix Agent.

Este servidor serve como demonstração do monitoramento via Zabbix, e para isso, por meio do Zabbix Agent instalado, são enviadas informações deste servidor para o servidor Zabbix onde é realizado monitoramento e exibição de informações tais como: consumo de armazenamento, CPU e memória.

A figura 4.10 exibe uma aplicação Web (HTML, CSS, JavaScript) desenvolvida na disciplina Programação para Dispositivos Móveis ministrado pelo professor Me. Felipe Maciel Rodrigues.

Figura 4.10 – Aplicação Web



Fonte: Elaborada pelo autor, 2023

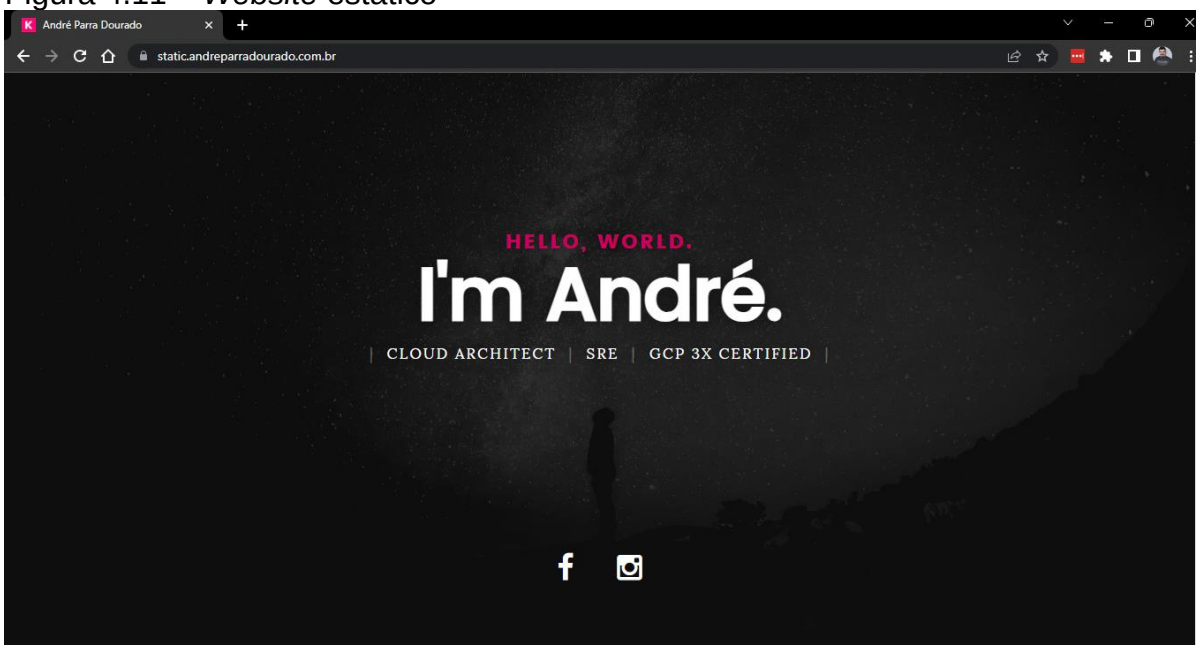
4.4.2 Website estático com bucket e load balancer HTTPS

A provisão de um *website* estático demonstra uma das possibilidades da *cloud*. Quando se trata de *website* estático, uma das possibilidades existentes na Google Cloud Platform é a hospedagem dos arquivos em um *bucket* de *backend* que receberá requisições que passam por um *load balancer*.

Realizou-se o *upload* dos arquivos do *website* para um *bucket* com a respectiva nomenclatura do *website* e então configurou-se um *load balancer* HTTPS que redireciona as requisições para o *bucket* configurado como *bucket* de *backend.* Deste modo, o *load balancer* está apto a efetuar o redirecionamento de acessos para o *bucket* com os arquivos do *website*.

A figura 4.11 demonstra uma *homepage* estática que utiliza HTML, CSS e JavaScript provisionada por meio da assossiação entre *bucket* e *load balancer.*

Figura 4.11 – *Website* estático



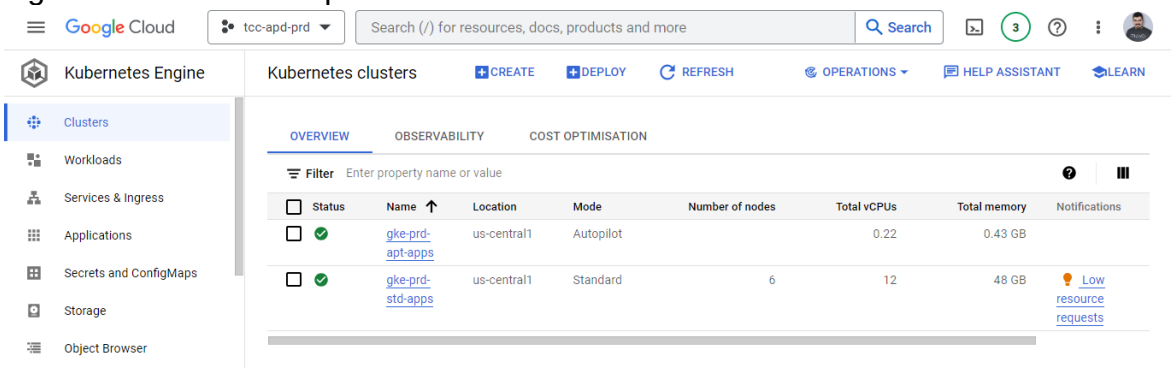
Fonte: Elaborada pelo autor, 2023

4.4.3 Cluster autopilot Kubernetes (autogerenciado)

Por meio do GKE foi implementado um *cluster autopilot* para aplicações diversas. No *cluster autopilot* o Google detém toda a gerência, *control plane* e *worker nodes*, restando neste caso apenas a função de *deploy* de aplicações para o consumidor.

A figura 4.12 é referente ao painel de *clusters* do Google Kubernetes Engine (GKE), onde estão localizados todos os *clusters* provisionados.

Figura 4.12 – *Clusters* provisionados no GKE



Fonte: Elaborada pelo autor, 2023

4.4.4 Cluster standard Kubernetes

Por meio do GKE foi implementado um *cluster standard,* e associadamente um *node pool* não preemptivo, para aplicações diversas. No modelo *standard* o Google não detém a gerência dos *worker nodes*, detém apenas do *control plane*, sendo necessário a criação, configuração e administração de *node pools* por parte do usuário.

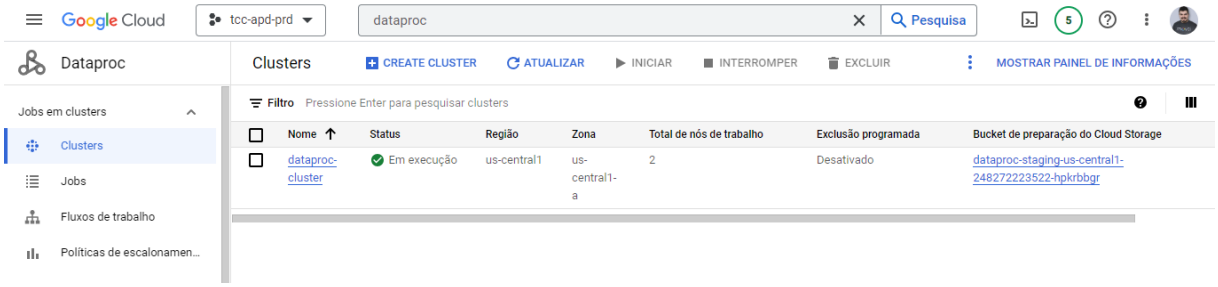
O modelo *standard* possui mais autonomia e menos restrições do que o *autopilot* demandando, consequentemente, mais conhecimento e responsabilidades do usuário.

4.4.5 Cluster Dataproc com ecossistema Hadoop

Implementou-se um *cluster* com ecossistemas Hadoop embutido, via Google Dataproc, utilizado comumente para funções como banco de dados distribuído para Big Data (Hbase), *data warehouse* (Hive), processamento de dados com Spark, entre outras funções.

A figura 4.13 exibe o *cluster* provisionado para o ambiente de produção, além disso, contém informações como região, zona, total de *nodes* do *cluster* e o *bucket* utilizado para armazenamento de arquivos temporários.

Figura 4.13 – *Clusters* Dataproc



Fonte: Elaborada pelo autor, 2023

4.5 PROVISIONAMENTO E DEMONSTRAÇÃO

O provisionamento de toda a infraestrutura inicia-se com o comando “deploy”, sendo este um *alias* para a execução do *script* deploy.sh.

Este *script* contém em suas primeiras linhas a definição de váriaveis utilizadas em seus comandos. Após isso, atualiza os módulos utilizados no projeto de gerência e então avança para o provisionamento dos recursos. Finalizado o provisionamento da infraestrutura do ambiente de gerência, inicia-se a atualização dos módulos do ambiente de produção e por fim prossegue para o provisionamento dos recursos deste ambiente como pode se ver ilustrado na figura 4.14.

Figura 4.14 - Início do script deploy.sh

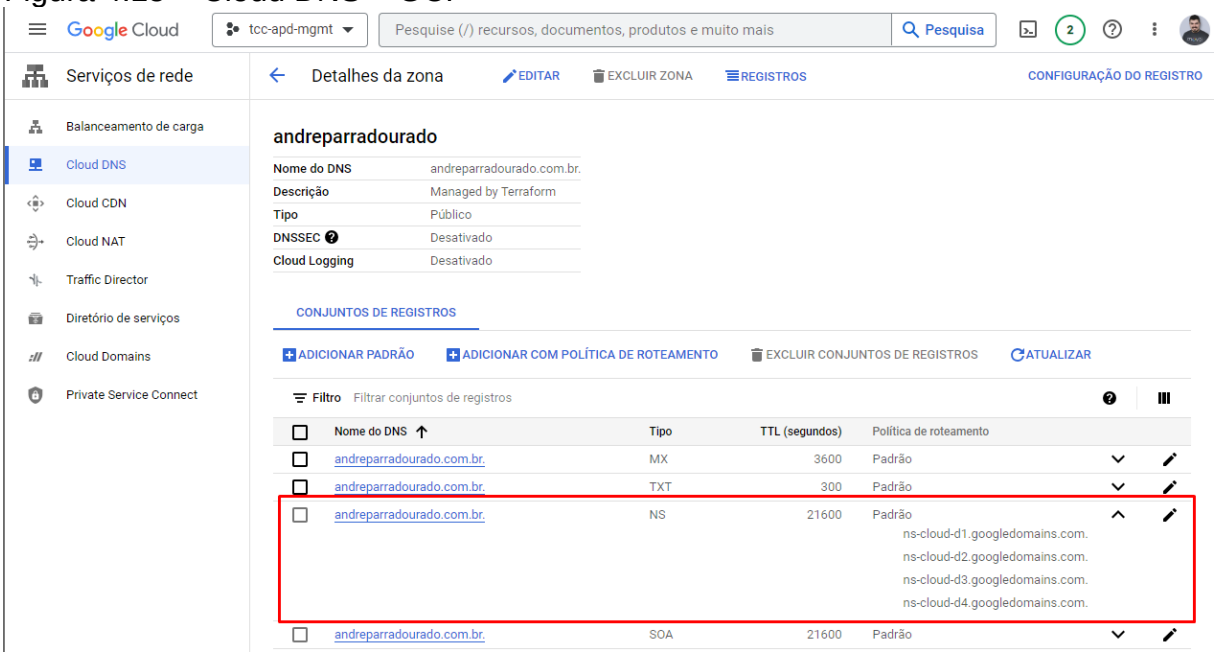
files > script > $ deploy.sh

```
1   #!/bin/bash
2
3   MGMT_PROJECT_ID="tcc-apd-mgmt"
4   MGMT_GKE_CLUSTER_NAME="gke-mgmt-olympus"
5   MGMT_GKE_LOCATION="us-central1"
6   MGMT_DNS_SHORT="andreparradourado"
7   MGMT_DNS_FULL="andreparradourado.com.br"
8   INGRESS_NGINX_VERSION="4.0.18"
9   CERT_MANAGER_VERSION="v1.11.0"
10  RANCHER_VERSION="2.7.3"
11  BOOTSTRAP_PW="admin"
12  SSL_EMAIL="andre.dourado@movti.com.br"
13
14  cd /root/terraform/environments/mgmt/
15  terraform init -upgrade
16  terraform apply -auto-approve
17  cd /root/terraform/environments/prd/
18  terraform init -upgrade
19  terraform apply -auto-approve
```

Fonte: Elaborado pelo autor, 2023

Em paralelo ao provisionamento do ambiente de produção, realiza-se o acesso ao Cloud DNS no console da GCP, do projeto de gerência, para que sejam coletados os registros NS (figura 4.15). Esses registros contém os servidores responsáveis por receber as requisições enviadas para o domínio utilizado no projeto.

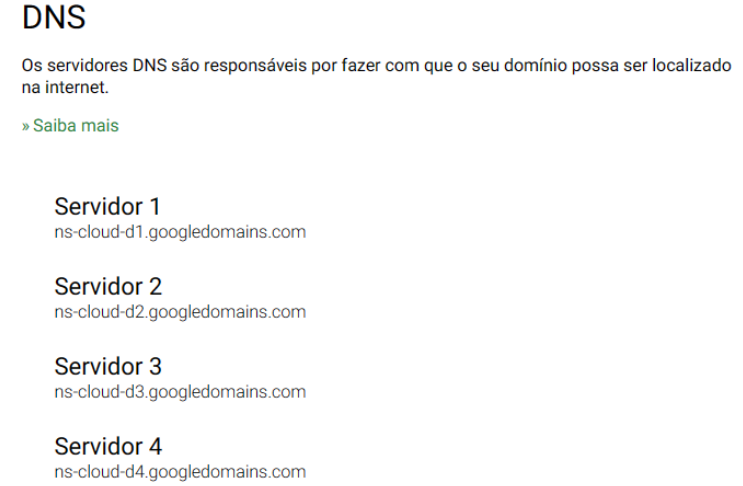
Figura 4.15 – Cloud DNS – GCP



Fonte: Elaborado pelo autor, 2023

Em seguida atualiza-se os servidores do Google - fornecidos no Cloud DNS - no Registro BR (figura 4.16) para que as requisições sejam redirecionadas para esta infraestrutura.

Figura 4.16 – Registro BR



Fonte: Elaborado pelo autor, 2023

Uma vez finalizado o provisionamento do ambiente de gerência, é executado um comando “gcloud” que aponta os comandos futuros para o ambiente de gerência e em sequência um comando responsável por enviar as credenciais de acesso, viabilizando a comunicação com o *cluster* GKE de gerência, ilustrados na figura 4.17.

Figura 4.17 - Armazenamento de endereço IP público reservado

```
21  cd /root/terraform/environments/mgmt/
22  gcloud config set project "$MGMT_PROJECT_ID"
23  gcloud container clusters get-credentials "$MGMT_GKE_CLUSTER_NAME" --location="$MGMT_GKE_LOCATION"
```

Fonte: Elaborado pelo autor, 2023

Concedidas as credenciais, o próximo comando executado (figura 4.18) tem como objetivo buscar o endereço IP público reservado no ambiente de gerência para que ele seja armazenado em uma variável e então atribuído como ponto de acesso do Rancher.

Figura 4.18 - Armazenamento de endereço IP público reservado

```
25  EXTERNAL_IP=$(gcloud dns record-sets describe rancher."$MGMT_DNS_FULL"
    --type=A --zone="$MGMT_DNS_SHORT" | awk 'NR == 2 {print $4}')
```

Fonte: Elaborado pelo autor, 2023

A partir desse momento, são executados três comandos Helm em sequência. Estes comandos são responsáveis pela preparação do *cluster* de gerência e instalação do Rancher no mesmo, pois este será utilizado para monitoramento do *cluster* Kubernetes do ambiente de produção.

A figura 4.19 contém o primeiro comando, responsável pela instalação do Ingress-nginx. O Ingress-nginx é um recurso de *load balancer* que expõe um endereço IP para receber as requisições da User Interface (UI) e API do Rancher.

Figura 4.19 - Comando Helm – Instalação Ingress-nginx

```
29  helm upgrade --install ingress-nginx ingress-nginx/ingress-nginx \
30    --namespace ingress-nginx \
31    --create-namespace \
32    --version "$INGRESS_NGINX_VERSION" \
33    --set controller.service.type=LoadBalancer \
34    --set controller.service.loadBalancerIP="$EXTERNAL_IP"
```

Fonte: Elaborado pelo autor, 2023.

O segundo comando executado, representado na figura 4.20, é responsável pela instalação do Cert-manager, aplicação responsável pela obtenção, renovação e utilização de certificados SSL/TLS dentro do *cluster.*

Figura 4.20 - Comando Helm – Instalação Cert-manager

```
36  helm upgrade --install cert-manager jetstack/cert-manager \
37    --namespace cert-manager \
38    --create-namespace \
39    --version "$CERT_MANAGER_VERSION" \
40    --set installCRDs=true
```

Fonte: Elaborada pelo autor, 2023

O terceiro, e último, comando executado é responsável pela instalação do Rancher, ilustrado na figura 4.21.

Figura 4.21 - Comando Helm – Instalação Rancher

```
42  helm upgrade --install rancher rancher-stable/rancher \
43      --namespace cattle-system \
44      --create-namespace \
45      --version "$RANCHER_VERSION" \
46      --set hostname=rancher."$MGMT_DNS_FULL" \
47      --set bootstrapPassword="$BOOTSTRAP_PW" \
48      --set ingress.ingressClassName=nginx \
49      --set ingress.tls.source=letsEncrypt \
50      --set letsEncrypt.email="$SSL_EMAIL" \
51      --set letsEncrypt.ingress.class=nginx
```

Fonte: Elaborado pelo autor, 2023

Neste ponto, restam apenas as poucas configurações manuais para que o ambiente esteja pronto para o uso, sendo elas: a finalização da instalação do Zabbix Server, do Rancher, e por fim a importação do *cluster* de produção no Rancher.

Em testes realizados, o tempo médio para que todo o ambiente esteja totalmente funcional foi de trinta minutos.

5 CONSIDERAÇÕES FINAIS

O tempo de resposta as variações de mercado, tanto para projetos já em execução quanto para novos, é crucial para o sucesso dos empreendimentos tanto quanto a qualidade da entrega em si. Antes das facilidades advindas da *cloud computing* as empresas possuíam menos, porém mais laboriosas e custosas, possibilidades de atender as demandas de infraestrutura de poder computacional.

Com a acessibilidade e popularização da *cloud computing* nota-se que além de ser uma questão econômica e de tempo, a cultura da transformação digital, aliada as tecnologias de vanguarda, é uma questão de sobrevivência em um mercado altamente volátil. Nesse contexto, surgem as ferramentas IaC com a proposta de tornar a implantação de infraestruturas um processo automatizado, com várias versões e reutilizável para que estas oportunidades sejam atendidas dentro dos termos anteriormente mencionados.

Neste projeto é provisionada uma infraestrutura capaz de suportar desde aplicações monolíticas quanto em *containers* que, após codificada, pode ser gerenciada de maneira prática e desvinculada de requisitos físicos. Ademais, os requisitos físicos podem ser impeditivos, a citar, a construção de um data center on-premises.

REFERÊNCIAS BIBLIOGRÁFICAS

DOCKER. **What is a Container? | Docker**. [S.l.]. Docker, 2023. Disponível em: https://www.docker.com/resources/what-container/. Acesso em: 28 mai. 2023.

FORTINET. **What is a Colocation Data Center? Types & Benefits | Fortinet**. [S.l.]. Fortinet, 2023. Disponível em: https://www.fortinet.com/resources/cyberglossary/colocation-data-center. Acesso em: 23 mai. 2023.

GITLAB. **What is Cloud native? | GitLab**. [S.l.]. GitLab, 2023. Disponível em: https://about.gitlab.com/topics/cloud-native/. Acesso em: 24 mai. 2023.

GOOGLE. **About Cloud Storage buckets | Google Cloud**. [S.l.]. Google, 2023a. Disponível em: https://cloud.google.com/storage/docs/buckets. Acesso em: 6 mar. 2023.

GOOGLE. **Access control best practices | Cloud Storage | Google Cloud**. [S.l.]. Google, 2023b. Disponível em: https://cloud.google.com/storage/docs/access-control/best-practices-access-control. Acesso em: 6 mar. 2023.

GOOGLE. **O que é computação em nuvem? | Google Cloud**. [S.l.]. Google Cloud, 2023c. Disponível em: https://cloud.google.com/learn/what-is-cloud-computing?hl=pt-br. Acesso em: 23 mai. 2023.

GOOGLE. **PaaS vs IaaS vs SaaS: What's the difference? | Google Cloud**. [S.l.]. Google Cloud, 2023d. Disponível em: https://cloud.google.com/learn/paas-vs-iaas-vs-saas. Acesso em: 1 jun. 2023.

GOOGLE. **Shared VPC | Google Cloud**. [S.l.]. Google Cloud, 2023e. Disponível em: https://cloud.google.com/vpc/docs/shared-vpc. Acesso em: 3 jun. 2023.

GOOGLE. **Visão geral do Google Cloud**. [S.l.]. Google Cloud, 2023f. Disponível em: https://cloud.google.com/docs/overview?hl=pt-br. Acesso em: 6 mar. 2023.

GOOGLE. **VPC networks | Google Cloud. [S.l.]**. Google Cloud, 2023g. Disponível em: https://cloud.google.com/vpc/docs/vpc. Acesso em: 3 jun. 2023.

GOOGLE. **What Is a Public Cloud? | Google Cloud**. [S.l.]. Google Cloud, 2023h. Disponível em: https://cloud.google.com/learn/what-is-public-cloud. Acesso em: 1 jun. 2023.

HASHICORP. **Resources - Configuration Language | Terraform | HashiCorp Developer**. [S.l.]. Hashicorp, 2023a. Disponível em: https://developer.hashicorp.com/terraform/language/resources/syntax. Acesso em: 18 jun. 2023.

HASHICORP. **State | Terraform | HashiCorp Developer**. [S.l.]. HashiCorp, 2023b. Disponível em: https://developer.hashicorp.com/terraform/language/state. Acesso em: 6 mar. 2023.

HASHICORP. **What is Terraform | Terraform | HashiCorp Developer**. What is Terraform?. [S.l.]. HashiCorp, 2023c. Disponível em: https://developer.hashicorp.com/terraform/intro. Acesso em: 22 mai. 2023.

HASHICORP. **What is Infrastructure as Code with Terraform? | Terraform | HashiCorp Developer**. [S.l.]. Hashicorp, 2023d. Disponível em: https://developer.hashicorp.com/terraform/tutorials/aws-get-started/infrastructure-as-code. Acesso em: 6 mar. 2023.

HELM. **Helm**. [S.l.]. Helm, 2023. Disponível em: https://helm.sh/. Acesso em: 30 mai. 2023.

HUGHES L.; KASUNIC M.; SWEENEY D. **Planning and Design Considerations for On-Premises Computing Environments**. CARNEGIE-MELLON UNIV PITTSBURGH PA; 2021. Disponível em: https://apps.dtic.mil/sti/pdfs/AD1138252.pdf. Acesso em: 06 jun. 2023.

IBM. **What is FaaS (Function-as-a-Service)? | IBM**. [S.l.]. IBM, 2023. Disponível em: https://www.ibm.com/topics/faas. Acesso em: 5 abr. 2023.

KRAUS, S.; JONES, P.; KAILER, N.; WEINMANN, A.; CHAPARRO-BANEGAS, N.; ROIG-TIERNO, N. (2021). **Digital Transformation: An Overview of the Current State of the Art of Research**. SAGE Open, 11(3). Disponível em: https://doi.org/10.1177/21582440211047576. Acesso em: 06 jun. 2023.

KUBERNETES. **Command line tool (kubectl) | Kubernetes**. [S.l.]. Kubernetes, 2023a. Disponível em: https://kubernetes.io/docs/reference/kubectl/. Acesso em: 30 mai. 2023.

KUBERNETES. **O que é Kubernetes? | Kubernetes**. [S.l.]. The Linux Foundation, 2023b. Disponível em: https://kubernetes.io/pt-br/docs/concepts/overview/what-is-kubernetes/. Acesso em: 23 mai. 2023.

MICROSOFT. **Containers vs. virtual machines | Microsoft Learn**. [S.l.]. Microsoft, 2023a. Disponível em: https://learn.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm. Acesso em: 18 jun. 2023.

MICROSOFT. **Environments - Cloud Adoption Framework | Microsoft Learn**. [S.l.]. Microsoft, 2023b. Disponível em: https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/considerations/environments. Acesso em: 18 jun. 2023.

NUTANIX. **What is DBaaS (Database-as-a-Service) | Nutanix**. [S.l.]. Nutanix, 2023. Disponível em: https://www.nutanix.com/info/what-is-dbaas. Acesso em: 5 abr. 2023.

OBJECTIVE. **Sistemas Legados: entenda o que é e quando trocar - Objective**. [S.l.]. Objective, 2023. Disponível em: https://www.objective.com.br/insights/sistemas-legados/. Acesso em: 24 mai. 2023.

ORACLE. **O que é Multicloud? | Oracle Brasil**. [S.l.]. Oracle, 2023. Disponível em: https://www.oracle.com/br/cloud/multicloud/what-is-multicloud/. Acesso em: 23 mai. 2023.

QI NETWORK. **Escalabilidade da nuvem: flexibilidade no armazenamento de dados**. [S.l.]. Qi Network, 2023. Disponível em: https://blog.qinetwork.com.br/escalabilidade-da-nuvem-flexibilidade-no-armazenamento-de-dados/. Acesso em: 24 mai. 2023.

RANCHER. **Enterprise Kubernetes Management | Rancher**. [S.l.]. Rancher, 2023. Disponível em: https://www.rancher.com/. Acesso em: 30 mai. 2023.

RASHID. A.; CHATURVEDI. A. **Cloud Computing Characteristics and Services: A Brief Review.** 2019. Disponível em: https://www.ijcseonline.org/full_paper_view.php?paper_id=3680. Acesso em: 04 jun. 2023

RED HAT. **O que são microsserviços?**. [S.l.]. Red Hat, 2023. Disponível em: https://www.redhat.com/pt-br/topics/microservices/what-are-microservices. Acesso em: 22 mai. 2023.

RED HAT. **What is CaaS?**. [S.l.]. Red Hat, 2020. Disponível em: https://www.redhat.com/en/topics/cloud-computing/what-is-caas. Acesso em: 5 abr. 2023.

SRINIVASAN. A.; MD. A. Q.; VARADARAJAN. V. **Era of Cloud Computing: A New Insight to Hybrid Cloud.** 2023. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050915005608?via%3Dihub. Acesso em: 04 jun. 2023.

SUSE. **On-Premises | SUSE Defines**. [S.l.]. SUSE, 2023. Disponível em: https://www.suse.com/suse-defines/definition/on-premises/. Acesso em: 23 mai. 2023.

TANENBAUM, A. S.; BOS, H. **Sistemas Operacionais Modernos**. 4. ed. São Paulo: Pearson Education do Brasil, 2016. 758 p.

VMWARE. **O que é um data center? | Glossário da VMware | BR**. [S.l.]. VMware, 2023a. Disponível em: https://www.vmware.com/br/topics/glossary/content/data-center.html. Acesso em: 23 mai. 2023.

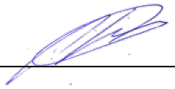
VMWARE. **What is Cloud Elasticity? | VMware Glossary**. [S.l.]. VMware, 2023b. Disponível em: https://www.vmware.com/topics/glossary/content/cloud-elasticity.html#:~:text=Cloud%20Elasticity%20is%20the%20property,changing%20demands%20of%20an%20organization.. Acesso em: 24 mai. 2023.

Página de assinaturas

Julio Lieira
080.769.668-41
Signatário

Felipe Rodrigues
345.484.558-09
Signatário

Adriano Bezerra
345.526.648-75
Signatário



Anderson Pazin
264.548.978-85
Aprovar

HISTÓRICO

22 jun 2023 19:36:45 — **Anderson Pazin** criou este documento. (E-mail: anderson.pazin@fatec.sp.gov.br, CPF: 264.548.978-85)

22 jun 2023 19:36:46 — **Anderson Pazin** *(E-mail: anderson.pazin@fatec.sp.gov.br, CPF: 264.548.978-85)* visualizou este documento por meio do IP 177.95.140.85 localizado em Sorocaba - Sao Paulo - Brazil

26 jun 2023 19:08:01 — **Anderson Pazin** *(E-mail: anderson.pazin@fatec.sp.gov.br, CPF: 264.548.978-85)* aprovou este documento por meio do IP 177.95.140.85 localizado em Sorocaba - Sao Paulo - Brazil

22 jun 2023 21:30:12 — **Julio Fernando Lieira** *(E-mail: julio.lieira3@fatec.sp.gov.br, CPF: 080.769.668-41)* visualizou este documento por meio do IP 189.126.178.174 localizado em Lins - Sao Paulo - Brazil

22 jun 2023 21:32:03 — **Julio Fernando Lieira** *(E-mail: julio.lieira3@fatec.sp.gov.br, CPF: 080.769.668-41)* assinou este documento por meio do IP 189.126.178.174 localizado em Lins - Sao Paulo - Brazil

26 jun 2023 15:09:17 — **Adriano Bezerra** *(E-mail: adriano.bezerra2@fatec.sp.gov.br, CPF: 345.526.648-75)* visualizou este documento por meio do IP 179.247.230.226 localizado em Brazil

26 jun 2023 15:09:22 — **Adriano Bezerra** *(E-mail: adriano.bezerra2@fatec.sp.gov.br, CPF: 345.526.648-75)* assinou este documento por meio do IP 179.247.230.226 localizado em Brazil

23 jun 2023 11:48:45 — **Felipe Maciel Rodrigues** *(E-mail: felipe.rodrigues30@fatec.sp.gov.br, CPF: 345.484.558-09)* visualizou este documento por meio do IP 201.182.122.14 localizado em Lins - Sao Paulo - Brazil

23 jun 2023 11:48:55 **Felipe Maciel Rodrigues** *(E-mail: felipe.rodrigues30@fatec.sp.gov.br, CPF: 345.484.558-09)* assinou este documento por meio do IP 201.182.122.14 localizado em Lins - Sao Paulo - Brazil