



**FACULDADE DE TECNOLOGIA DE AMERICANA “MINISTRO RALPH BIASI”
Curso Superior de Tecnologia em Segurança da Informação**

Cindy Sanches Lima
Paula Mezzomo

**ESTUDO SOBRE COMPUTAÇÃO EM NUVEM E SUAS APLICAÇÕES
COM KUBERNETES**

Americana, SP
2022

**FACULDADE DE TECNOLOGIA DE AMERICANA “MINISTRO RALPH BIASI”
Curso Superior de Tecnologia em Segurança da Informação**

Cindy Sanches Lima
Paula Mezzomo

**ESTUDO SOBRE COMPUTAÇÃO EM NUVEM E SUAS APLICAÇÕES
COM KUBERNETES**

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Segurança da Informação, sob a orientação da Profa. Dra. Maria Cristina Aranda

Área de concentração: Segurança da Informação

Americana, SP.
2022


Cindy Sanches Lima
Paula Mezzomo

**ESTUDO SOBRE COMPUTAÇÃO EM NUVEM E SUAS APLICAÇÕES COM
KUBERNETES**

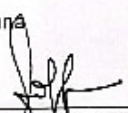
Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Curso Superior de Tecnologia em Segurança da Informação pelo Centro Paula Souza - FATEC Faculdade de Tecnologia de Americana - Ralph Biasi.
Área de concentração: Segurança da Informação

Americana, 20 de junho de 2022

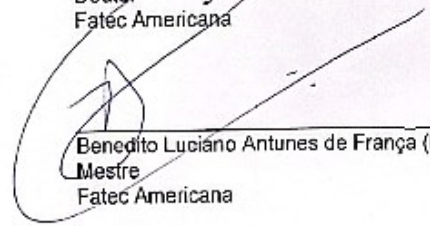
Banca Examinadora:



Maria Cristina Aranda (Presidente)
Doutora
Fatec Americana



Renato Kraide Soffner (Membro)
Doutor
Fatec Americana



Benedito Luciano Antunes de França (Membro)
Mestre
Fatec Americana

DEDICATÓRIA

Aos nossos familiares, aos nossos colegas de curso Gabriele Arbertavicius, Marcelo Luvezutto Júnior, Mario Octavio Cordeiro Carmesini, que assim como nós encerram uma etapa da vida acadêmica, a nossa orientadora Maria Cristina, aos nossos professores, ao Gean Carlos de Oliveira e Gustavo Henrique Geraldo.

RESUMO

As tecnologias estão crescendo exponencialmente nos últimos anos, evoluindo de maneira rápida, mudando o mundo dos negócios e a vida das pessoas. Nesse contexto, a Nuvem, ganha cada dia mais adeptos. Dentro dela é possível processar, de maneira rápida, eficaz e com baixo custo as mais diversas tecnologias modernas, tais como *Blockchain*, Inteligência Artificial, Internet das Coisas, entre outras, trazendo um poder sem igual para os negócios. Com o intuito de compartilhar conhecimento, levando em conta que poucas pessoas entendem o funcionamento e o poder da Nuvem, a falta de profissionais qualificados no mercado, esse trabalho foi escrito para esclarecer os conceitos e mostrar serviços que podem ser utilizados independentemente do provedor. Será possível entender o que é a Nuvem, como ela funciona, prós e contras, segurança, virtualização (considerado o coração da nuvem), Kubernetes, microsserviços, redes e outros tópicos importantes. Para facilitar o entendimento foram acrescentadas figuras representativas e um capítulo para mostrar na prática, utilizando Kubernetes e Docker, os conceitos essenciais sobre a nuvem. Objetivo principal da prática foi demonstrar os conceitos sobre nuvem usando ferramentas que possuem um padrão de comandos a serem executados e demonstrar os resultados obtidos após execução de cada etapa. Todo o trabalho foi baseado em pesquisas feitas em livros, *site* sobre o tema e cursos realizados durante a elaboração do mesmo. Também, foram utilizadas documentações do Instituto Nacional de Padrões e Tecnologia ou NIST, uma agência governamental não regulatória da administração de tecnologia do departamento de comércio dos Estados Unidos. Todas as referências podem ser encontradas no capítulo de Bibliografia. Ao final, o leitor com conhecimentos básicos na área de tecnologia, entenderá o que é a nuvem.

Palavras Chave: Cloud Computing; Segurança da Informação; Kubernetes

ABSTRACT

Technologies are growing exponentially in recent years, evolving rapidly, changing the business world and people's lives. In this context, the Cloud has been gaining more followers every day. Within it, it is possible to process, quickly, effectively and at low cost the most diverse modern technologies, such as Blockchain, Artificial Intelligence, Internet of Things, among others, bringing unparalleled power to business. In order to share knowledge, taking into account that few people understand the operation and power of the Cloud, the lack of qualified professionals in the market, this work was written to clarify the concepts and show services that can be used regardless of the provider. It will be possible to understand what the Cloud is, how it works, pros and cons, security, virtualization (considered the heart of the cloud), Kubernetes, microservices, networks and other important topics. To facilitate understanding, representative figures and a chapter were added to show in practice, using Kubernetes and Docker, the essential concepts about the cloud. In this step, the main objective was to demonstrate the cloud concepts using tools that have a pattern of commands to be executed and demonstrate the results obtained after the execution of each step. All work was based on research done in books, website on the subject and courses taken during its elaboration. Also, documentation from the National Institute of Standards and Technology or NIST, a non-regulatory government agency of the technology administration of the US Department of Commerce, was used. All references can be found in the Bibliography chapter. In the end, the reader with basic knowledge in the area of technology will understand what the cloud is.

Keywords: *Computação em Nuvem; Segurança da Informação; Kubernetes*

LISTA DE FIGURAS

Figura 1. IBM System/360 Modelo 91	12
Figura 2. Representação do Sobre Demanda – “Self-Service”	13
Figura 3. Representação do Amplo Acesso a Rede.....	14
Figura 4. Agrupamento de Recursos.....	15
Figura 5. Rápida Elasticidade.....	16
Figura 6. Serviços Mensuráveis	16
Figura 7. Pirâmide de Nível de Serviço	17
Figura 8. Comparação Entre os Modelos: Tradicional x Nuvem	19
Figura 9. Data Center no Chile.....	25
Figura 100. Regiões	26
Figura 11. Bare Metal X Hosted	31
Figura 12. Funcionamento do Kubernetes	34
Figura 13. Armazenamento Direct Attached.....	39
Figura 14. Armazenamento File Storage.....	40
Figura 15. Armazenamento Block Storage.....	41
Figura 16. Armazenamento Object Storage	42
Figura 17. Redes Privadas.....	43
Figura 18. Sub-rede e Aplicativos	44
Figura 19. Load Balancer	45
Figura 20. Aplicação em Microserviços	63
Figura 21. Módulos Básico do Kubernetes.....	71

LISTA DE TABELAS

Tabela 1. Mapa dos Padrões de Segurança	50
Tabela 2. Mapa dos Padrões de Segurança	51
Tabela 3. Mapa dos Padrões de Segurança	52

SUMÁRIO

INTRODUÇÃO	9
2. DEFINIÇÃO SOBRE NUVEM – CLOUD E O MERCADO	10
2.1. INTERNET	12
2.2. CARACTERÍSTICAS ESSENCIAIS DA NUVEM	13
2.2.1. Autoatendimento sob demanda (On-Demand Self-Service)	13
2.2.2. Amplo Acesso a rede (Broad Network Access)	14
2.2.3. Agrupamento de Recursos (Resource Pooling)	14
2.2.4. Rápida Elasticidade (Rapid Elasticity)	15
2.2.5. Serviços Mensuráveis (Measured Service)	16
2.3. MODELO DE SERVIÇOS	16
2.3.1. Software como Serviço	17
2.3.2. Plataforma como Serviço	17
2.3.3. Infraestrutura como Serviço	18
2.4. COMPARAÇÃO ENTRE O MODELO TRADICIONAL DE TECNOLOGIA E A NUVEM	18
2.5. NUVEM PRIVADA, PÚBLICA, HÍBRIDA E COMUNITÁRIA	19
2.5.1. Nuvem Privada	20
2.5.2. Nuvem Pública	20
2.5.3. Nuvem Híbrida	20
2.5.4. Nuvem Comunitária	20
2.6. O QUE DEVE SER CONSIDERADO PARA ADOÇÃO DA NUVEM	20
2.7. BENEFÍCIOS x RISCO DA ADOÇÃO DA NUVEM	21
2.8. NATIVO DE NUVEM	23
2.8.1. Passível de automação	23
2.8.2. Onipresente e flexível	23

2.8.3.	Resiliente e escalável	23
2.8.4.	Dinâmico	23
2.8.5.	Observável	24
2.8.6.	Distribuído	24
2.9.	INFRAESTRUTURA DA NUVEM	24
2.9.1.	VIRTUALIZAÇÃO, MÁQUINAS VIRTUAIS	27
2.9.1.1.	Tipo 1	27
2.9.1.2.	Tipo 2	28
2.9.2.	BARE METAL	28
2.9.3.	BARE METAL X MÁQUINAS VIRTUAIS	29
2.9.4.	CONTÊINERES	30
2.9.4.1.	BENEFÍCIOS DO CONTÊINER	31
2.9.5.	DOCKER, KUBERNETES E OS CONTÊINERES	32
2.9.6.	KUBERNETES	32
2.9.6.1.	ARQUITETURA KUBERNETES	33
2.9.6.1.1.	Pod	33
2.9.6.1.2.	Node	33
2.9.6.1.3.	Cluster	33
2.9.6.1.4.	Master	33
2.9.6.1.5.	Seguro	34
2.9.6.1.6.	Fácil de usar	34
2.9.6.1.7.	Extensível	34
2.9.6.2.	COMPONENTES	34
2.9.6.2.1.	API Server	34
2.9.6.2.2.	Etcd	35
2.9.6.2.3.	Kubelet	35

2.9.6.2.4. Container Runtime	35
2.9.6.2.5. Control Plane	35
2.9.6.2.6. Scheduler	35
2.9.6.2.7. Namespaces	35
2.9.6.2.8. NodePort	35
2.9.6.2.9. ClusterIP	35
2.9.7. MICROSERVIÇOS	36
2.9.7.1. CINCO CARACTERÍSTICAS DOS MICROSERVIÇOS	36
2.10. ARMAZENAMENTO	38
2.10.1. DIRECT ATTACHED	39
2.10.2. FILE STORAGE	39
2.10.3. BLOCK STORAGE	40
2.10.4. OBJECT STORAGE	42
2.11. REDE	43
3. SEGURANÇA	45
3.1. RESPONSABILIDADES	48
3.2. MAPA DOS PADRÕES DE SEGURANÇA	50
3.3. SEIS ETAPAS PARA UMA COMPUTAÇÃO EM NUVEM MAIS SEGURA	52
3.3.1. Aproveitar os Recursos de Segurança oferecidos pelas empresas de serviço em nuvem	52
3.3.2. Fazer inventários regulares do que mantém na nuvem	53
3.3.3. Não armazenar informações pessoais quando não for necessário	53
3.3.4. Considerar encriptografar dados raramente usados	53
3.3.5. Prestar atenção aos avisos confiáveis	54
3.3.6. A segurança é responsabilidade do cliente	54
4. NUVEM NA PRÁTICA	54

CONCLUSÃO	72
REFERÊNCIAS	73

INTRODUÇÃO

Em um mundo cada vez mais conectado, pessoas buscam a facilidade dos serviços *online* e a rápida conexão de qualquer parte do mundo, benefícios que podem ser trazidos por *Cloud*. O termo *Cloud*, também conhecido como Nuvem, abrange uma variedade de serviços que podem atender a muitas demandas tecnológicas de diversos modelos de negócios existentes, clientes e indústrias.

Nos últimos anos, principalmente com o crescimento de tecnologias que envolvem dados como Inteligência Artificial, Blockchain entre outras, o uso de *Cloud Computing* aumentou consideravelmente já que essas tecnologias demandam um grande uso de memória e velocidade dos computadores. A cada dia, mais pessoas passam a conhecer o termo e, muitos dos serviços da Nuvem já fazem parte do cotidiano da população mundial e seu uso até se tornou comum, como por exemplo, o serviço G-Mail.

Esse trabalho foi escrito com o objetivo de explicar o que é a Nuvem e fazer com que mais pessoas entendam seu funcionamento, desde que tenham um conhecimento prévio em tecnologia. Nele, serão apresentados desde conceitos básicos para o entendimento inicial até conceitos mais complexos como Kubernetes e padrões de segurança a serem seguidos.

Os conceitos aqui apresentados serão relacionados à Nuvem e não ao provedor onde cada empresa tem sua própria maneira de vender os seus produtos.

O trabalho foi estruturado em cinco capítulos, onde os iniciais trazem os conceitos básicos, uma pequena introdução sobre Internet já que os serviços na nuvem não existem sem a mesma, prós e contra de adotar a Nuvem entre outros temas relevantes e, no último capítulo, serão apresentados na prática o uso do Kubernetes para demonstrar os conceitos de disponibilidade, escalabilidade, rede, entre outros que serão mencionados ao longo do trabalho. Nele será possível visualizar os benefícios de, por exemplo, colocar uma aplicação na Nuvem e a rapidez com que os serviços podem ser escalados. Para facilitar o entendimento também foram utilizadas imagens representativas.

Ao final, o leitor será capaz de entender o que é a Nuvem e os serviços essenciais que podem ser encontrados em qualquer provedor.

Como escreveu Kahlil Gibran, escritor Libanês: “A perplexidade é o início do conhecimento.”

2. DEFINIÇÃO SOBRE NUVEM – CLOUD E O MERCADO

O conceito de nuvem, ou mais conhecido como *Cloud*, existe desde a década de 60 (aproximadamente), uma época que os computadores ocupavam salas enormes, fato este apresentado na Figura 1. Nessa época poucas pessoas tinham acesso ao processamento central e seu custo de fabricação e manutenção era elevado. Era mais viável para as empresas utilizarem a infraestrutura computacional fornecida por terceiros e depois, receber uma conta pelo tempo de processamento ou volume de recursos que foram utilizados, conhecidos pelos termos *pay-as-you-go* ou *pay-per-use*. Termos que permanecem vigente para a prestação de muitos serviços que são fornecidos atualmente pelos provedores de Nuvem.

O termo Computação em Nuvem, do inglês *Cloud Computing*, passou a ser popular em 1997 após uma palestra acadêmica realizada pelo professor Ramnath Chellappa (DELL INC, 2022), que atualmente é reitor, diretor acadêmico e professor de Sistemas de Informação e Gerenciamento de Operações na Emory – Goizueta Business School em Atlanta, Estados Unidos. Ele foi o primeiro a definir em um trabalho acadêmico o termo *Cloud Computing*, o que lhe concedeu um PhD na McCombs School of Business, universidade do Texas (EMORY UNIVERSITY'S GOIZUETA BUSSINESS SCHOOL, 2021).

Arundel e Domingus (2019, p. 26) definiram que “A idéia central da Nuvem é esta: em vez de comprar um computador, compramos computação”. O que faz voltar ao conceito criado na década de 60 onde a infraestrutura dos serviços era fornecida por um terceiro. E como menciona os mesmos autores “A palavra revolução significa “movimento circular” (ARUNDEL; DOMINGUS, 2019, p. 25), e a computação, de certo modo, voltou ao ponto que começou. Embora os computadores tenham adquirido muito mais potência ao longo dos anos.”

As pessoas e empresas que optam por usar os serviços da Nuvem não precisam se preocupar com a infraestrutura física, com os custos para comprar novos equipamentos, trocar os que estão obsoletos e custos com manutenção. Elas compram capacidade de processamento remoto, sistemas distribuídos, entre outros e, principalmente, compra-se tempo, pois segundo Arundel e Domingus (2019, p. 26) “... pode simplesmente comprar tempo do computador de outras pessoas e deixar que elas cuidem de sua escala, da manutenção e do *upgrade*”.

Considerando a revolução tecnológica que ocorreu nos últimos anos, a criação de computadores com alta capacidade de processamento, pode-se dizer que a Nuvem fornece um conjunto de recursos com características como rapidez, agilidade, acesso simplificado e global, fácil manutenção, recursos sob demanda, escalabilidade e elasticidade, termos que serão explorados ao longo desse trabalho.

Atualmente, a Nuvem é utilizada nas mais diversas áreas do dia a dia das pessoas e empresas, por exemplo, nas compras e interações *online*, para realizar transações bancárias, jogos, usar aplicativos e diversos outros usos que muitas vezes são considerados essenciais, como por exemplo, na saúde.

Segundo pesquisas da empresa GARTNER, INC. AND/OR ITS AFFILIATES (2020), líder mundial em pesquisa e aconselhamento para empresas, realizada em 2020 com a pandemia do COVID19, o crescimento da Nuvem em 2021 seria de 18.4% comparado a 2020: “A pandemia de COVID-19 confirmou os diferenciais e a proposta de valor das soluções em Nuvem”, afirma Sid Nag, Vice-Presidente de Pesquisa da Gartner e ainda acrescenta:

A capacidade de utilizarmos modelos Cloud escalonáveis para obter eficiência de custos e continuidade de negócios está impulsionando as organizações a acelerarem seus planos de Transformação Digital. Agora mais do que nunca, o aumento do uso de serviços em Nuvem Pública reforça o ‘novo normal’. (GARTNER, INC. AND/OR ITS AFFILIATES, 2020)

Ainda segundo essa pesquisa, é previsto que, em 2024, *Cloud* irá representar 14,2% dos gastos com tecnologia, contra os 9,1% registrados em 2020.

Estudo publicado pelo INTERNATIONAL DATA CORPORATION (2017) prevê que até 2025, a quantidade total de dados criados em todo o mundo aumentará para 163 *zettabytes* (onde um *zettabyte* é equivalente a um trilhão de *gigabytes*). E 30% desses dados serão informações em tempo real. (apud REINSEL; GANTZ; RYDNING, 2017)

Considerando a quantidade sem precedentes de dados produzidos diariamente e a capacidade de tomar decisões baseadas em dados cruciais para qualquer negócio, a computação em Nuvem se torna essencial para as empresas terem sucesso, sustentar e competir nos mercados atuais.

Uma estratégia de Nuvem, mais do que apenas uma estratégia de TI, é o componente central de qualquer estratégia de negócios hoje. Empresas que ainda não integraram ou não estão integrando a Nuvem em suas estratégias de negócio, correm o risco de não ter velocidade, agilidade, inovação e boa tomada de decisão.

Capacidades necessárias para serem competitivas, assim como sua capacidade de responder à disrupção digital.

Mell e Grance (2011, p. 2) definiu a computação em Nuvem:

A computação em nuvem é um modelo para permitir o acesso onipresente, conveniente e sob demanda à rede compartilhada, conjunto de recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que pode ser provisionado e liberado rapidamente com o mínimo de esforço de gerenciamento ou interação do provedor de serviços.

Figura 1. IBM System/360 Modelo 91



Fonte: Ortega, 2021

2.1. INTERNET

Segundo Torres (2022), autor de Livros e dono do *site* Clube do Hardware, a Internet é uma rede pública e mundial de computadores que também pode ser chamada de “Global Area Network” - GAN. Ela é formada por diversas redes interconectadas e redundantes, ou seja, quando um caminho está com algum tipo de problema pode ser usado um outro caminho para atingir o objetivo.

Cada rede que faz parte da infraestrutura da Internet é chamada de Sistema Autônomo.

A Internet é uma rede pública isso porque qualquer pessoa pode ter acesso, mas ela também pode ser privada, chamada de Intranet, quando, por exemplo, o dono da rede (empresa ou pessoa física) limita o acesso.

A Internet possui vários tipos de serviços como por exemplo, baixar arquivos (PDF, JPEG etc.), envio de *e-mail*, VoIP, videoconferência, jogos *online*, *web*, entre outros. Lembrando que Internet e *web* não são sinônimos: *web* é um serviço disponível na Internet. O encaminhamento de dados que ocorre entre redes é denominado roteamento.

A Internet permite a criação da Nuvem, pois um usuário pode acessar através da rede a infraestrutura de *hardware* e *software* sem saber onde está a localização deles. O que importa para o usuário é a localização do dispositivo, computador, *tablet*, celulares etc. que irá acessar a informação.

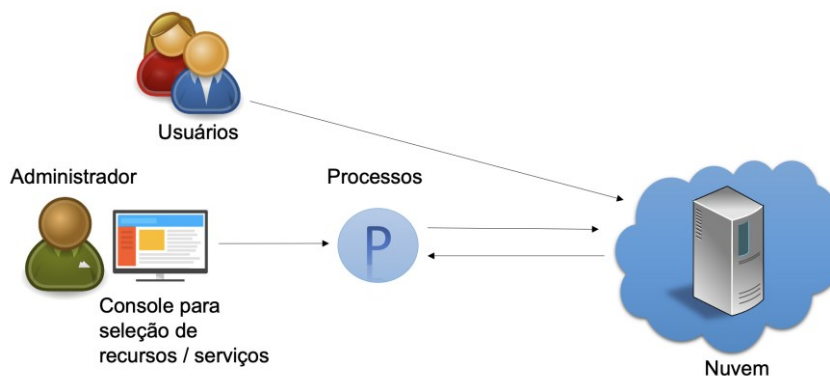
2.2. CARACTERÍSTICAS ESSENCIAIS DA NUVEM

Existem algumas características da Nuvem que são consideradas essenciais segundo Mell e Grande (2011, p. 2-3) que serão mencionadas a seguir:

2.2.1. Autoatendimento sob demanda (On-Demand Self-Service)

O consumidor acessa a plataforma de serviços da Nuvem, escolhe o que deseja utilizar, define a quantidade e processamento para uso sem a necessidade de interação com o atendimento do provedor. O pagamento é feito conforme o uso. A figura 2 mostra um administrador do sistema que tem acesso a plataforma de serviços, faz o controle dos mesmos e os usuários que utilizam o que lhes é disponibilizado.

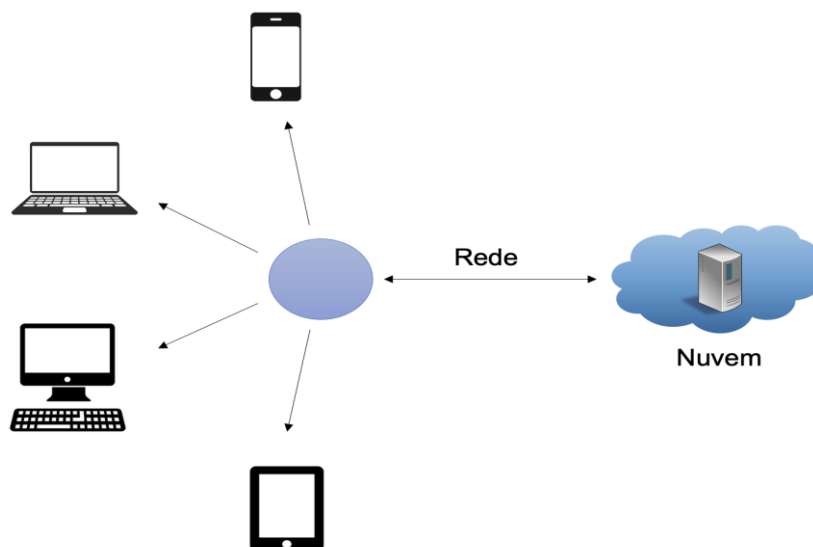
Figura 2. Representação do Sobre Demanda – “Self-Service”



2.2.2. Amplo Acesso a rede (Broad Network Access)

Os recursos estão na rede e podem ser acessados de qualquer plataforma como tablets, celulares, computadores etc. É necessário ter uma conexão com a rede para usar os serviços na Nuvem. Representação feita na figura 3.

Figura 3. Representação do Amplo Acesso a Rede

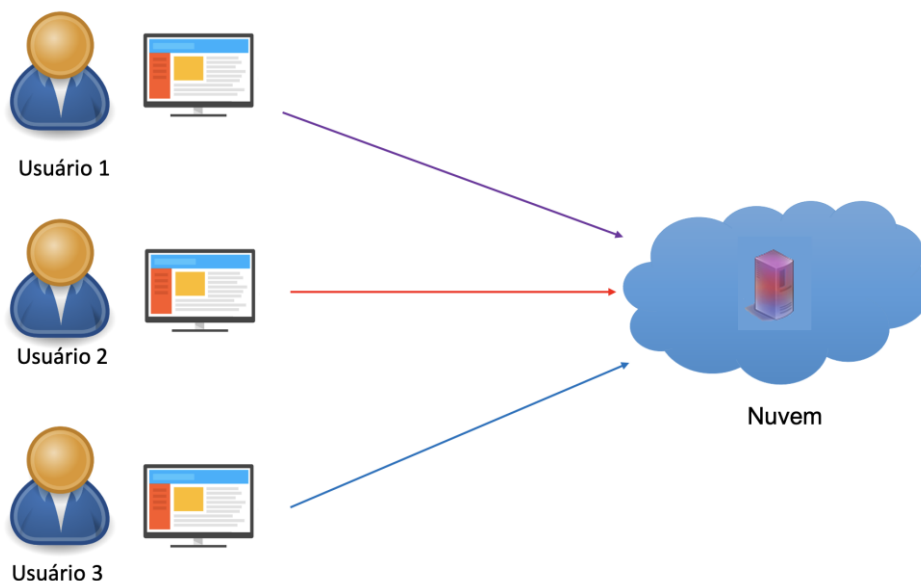


Fonte: adaptado de Corbett, 2022

2.2.3. Agrupamento de Recursos (Resource Pooling)

Os recursos que podem ser utilizados na Nuvem são reunidos geograficamente. O consumidor não sabe a real localização dos recursos e não tem controle sobre isso, o que ele consegue saber é o país, estado ou o nome do data center. Os recursos podem ser: memória, processamento, softwares etc. Nesse aspecto é importante a independência do local, que é um cliente não ter acesso ao dado do outro cliente, o que é chamado de multitenand. Na figura 4, os três usuários acessam os serviços da Nuvem de maneira independente.

Figura 4. Agrupamento de Recursos

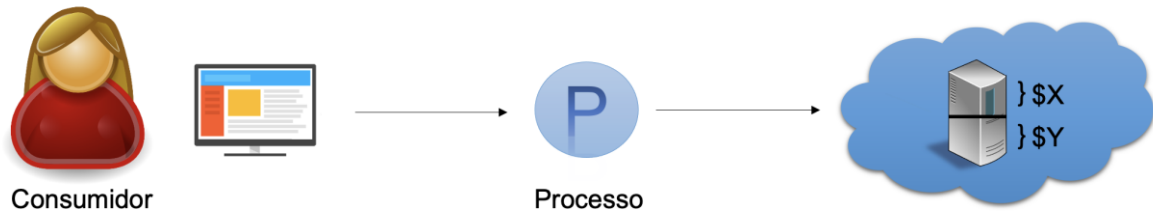


Fonte: adaptado de Corbett, 2022

2.2.4. Rápida Elasticidade (Rapid Elasticity)

A Nuvem traz ao consumidor a idéia de ser infinita, pois ele pode adquirir mais ou menos recursos conforme demanda. Por exemplo, um site de e-commerce hospedado na Nuvem tem uma quantidade de recurso (memória e processamento) disponibilizada para uma quantia de acessos ao mesmo. Em uma data específica, os produtos desse site são colocados em oferta, com isso há um aumento no uso da página. Esse aumento no acesso exige mais recursos. Na Nuvem, isso pode ser feito de maneira rápida e até mesmo pode ser configurado para que o recurso, de maneira automática, aumente quando a demanda for maior e diminua assim que a demanda for menor. Na figura 5, o consumidor possui x memória e pode adquirir mais y no aumento da demanda.

Figura 5. Rápida Elasticidade

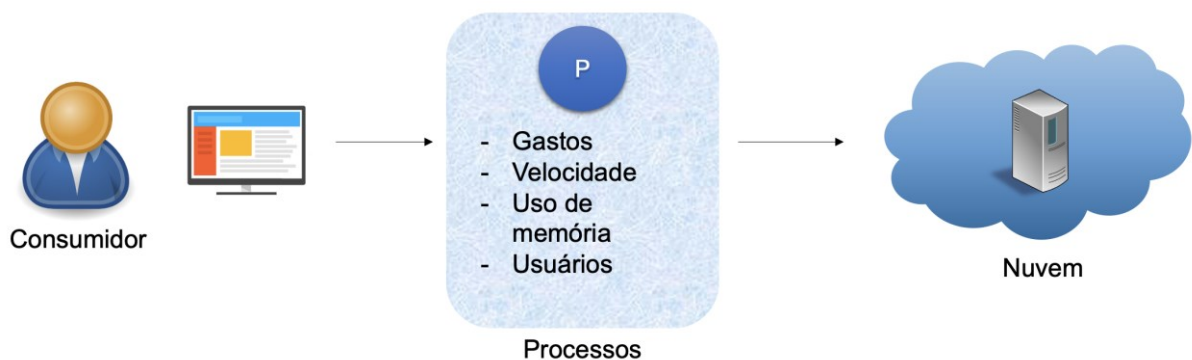


Fonte: adaptado de Corbett, 2022

2.2.5. Serviços Mensuráveis (*Measured Service*)

O uso dos serviços – recursos na Nuvem são medidos, monitorados, controlados e transparentes ao cliente. Conforme figura 6, ele possui um painel ou console para controle. Isso facilita o uso da Nuvem de acordo a produção de cada consumidor e ajuda o provedor a realizar a cobrança do uso dos recursos tais como armazenamento, largura de banda, processamento, contas de usuários ativos etc.

Figura 6. Serviços Mensuráveis

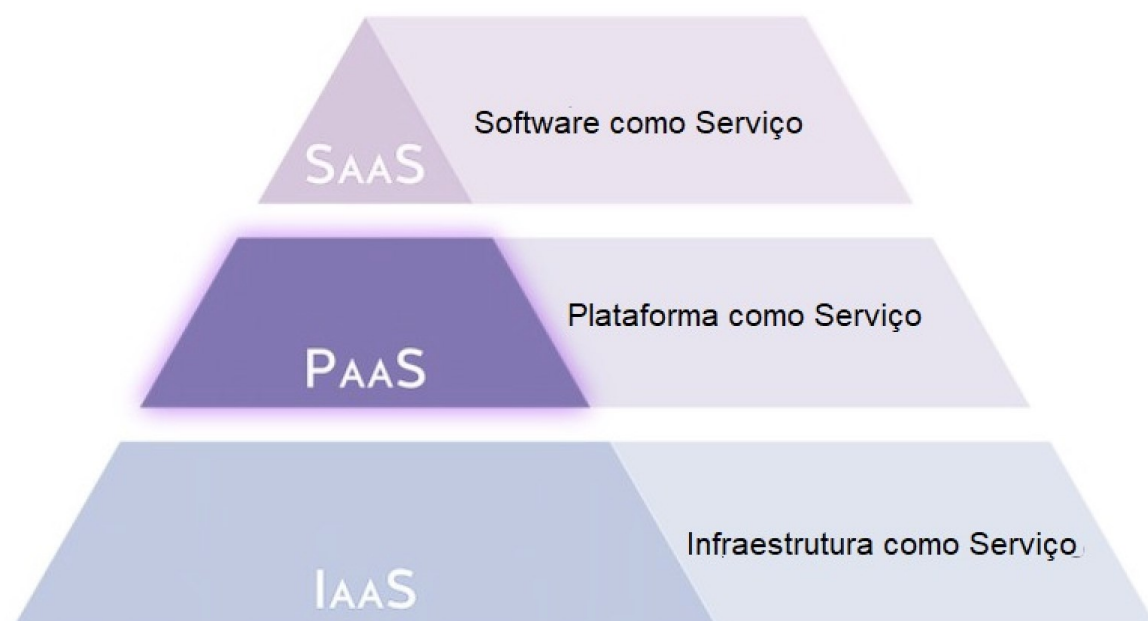


Fonte: adaptado de Corbett, 2022

2.3. MODELO DE SERVIÇOS

Existem três níveis de serviços fornecidos pela Nuvem: IaaS – *Infrastructure as a Service* ou Infraestrutura como Serviço, PaaS – *Platform as a Service* ou Plataforma como um serviço e SaaS – *Software as a Service* ou Software com serviço.

Figura 7. Pirâmide de Nível de Serviço



Fonte: adaptado de 4D DATA CENTRES LTD, 2020

Mell e Grande (2011, p. 2-3) define cada serviço:

2.3.1. Software como Serviço

O consumidor da nuvem utiliza os serviços fornecidos pelo provedor e pode acessá-los de vários dispositivos. Ele não controla a infraestrutura, isto é não controla redes, servidores, sistemas operacionais, armazenamento etc. Esse serviço pode ser, por exemplo, um serviço de e-mail baseado na web, Office365, Dropbox, etc.

2.3.2. Plataforma como Serviço

Nesse serviço o consumidor também não controla a infraestrutura, mas ele já pode criar ou adquirir aplicativos criados usando programação e pode definir as configurações para o ambiente que irá hospedar sua aplicação. Pode, por exemplo, hospedar um site web com produtos para venda. Nesse serviço há servidores, redes, armazenamento, sistema operacional, middlewares, banco de dados etc. O usuário é apenas responsável pelo código do aplicativo e sua manutenção.

2.3.3. Infraestrutura como Serviço

O consumidor já tem acesso ao processamento, armazenamento, redes e outros recursos de computação fundamentais para implementar e executar softwares “arbitrários”, sistemas e aplicações. Nessa parte ele pode, por exemplo, criar ou provisionar máquinas virtuais com os recursos de sua preferência e realizar configurações no servidor.

O formato de pirâmide, visto na figura 7, é explicado por Augusto (2012), diretor de Inovação e sócio-fundador da Qi Network

[...] como uma escalada tecnológica, em que uma organização vai ampliando os serviços utilizados na nuvem, partindo de ferramentas básicas de colaboração em SaaS, passando pela utilização de softwares em PaaS, até chegar à migração parcial ou completa da infraestrutura de TI para a nuvem (IaaS).

2.4. COMPARAÇÃO ENTRE O MODELO TRADICIONAL DE TECNOLOGIA E A NUVEM

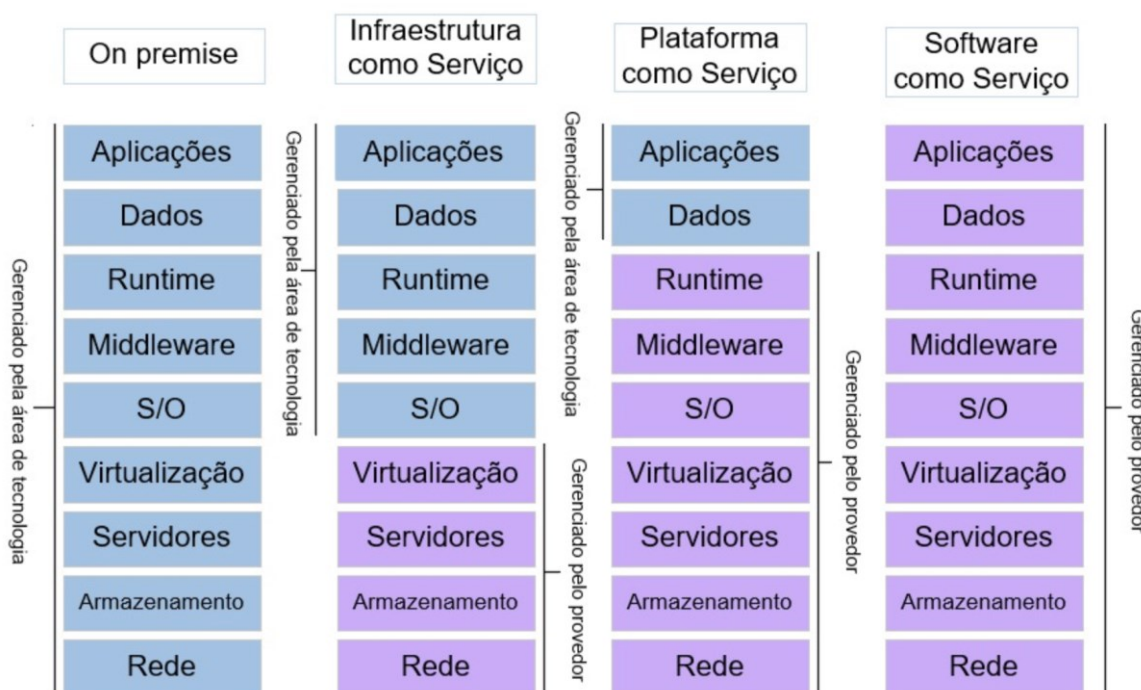
O *site* da Microsoft Azure apresenta a figura 8, que compara os níveis de serviço na Nuvem e a estrutura tradicional de tecnologia.

Na primeira coluna, *On Premises*, modelo tradicional, todos os itens estão destacados em azul, pois eles são de inteira responsabilidade da área de tecnologia da empresa que os mantém. Existe o cuidado com os dados, a virtualização através das VMs, as configurações dos serviços de rede etc. Em seguida, começam os modelos da Nuvem, e é possível visualizar o IaaS onde a parte azul continua sendo os serviços que são responsabilidade da área de tecnologia e a parte lilás, os serviços do provedor de Nuvem. Nesse modelo os serviços que podem ser encontrados é o de armazenamento em rede dos servidores, a virtualização do sistema operacional, e pode ser criado, por exemplo, um banco de dados.

Em PaaS, as opções de serviços fornecidas por um provedor aumentam e é de responsabilidade do cliente os dados e as aplicações. Nessa opção de nível de serviço já é possível utilizar serviços como fila de mensagens, gerenciar banco de dados e armazenamentos. No exemplo de um banco de dados não é necessário cuidar do sistema operacional e nem do gerenciamento, mas somente criar os dados dentro do banco.

Em SaaS todos os serviços são gerenciados pelo provedor de serviços. Existe um nível de configuração que pode ser feita pelo cliente, mas a pilha inteira é gerenciada por outra pessoa ou provedor. Um exemplo que pode ser mencionado é o G-Mail da Google, o aplicativo é implementado em um servidor em lugar não conhecido e não pode ser atualizado ou mudado.

Figura 8. Comparação Entre os Modelos: Tradicional x Nuvem



Fonte: adaptado de Stack247, 2015

No próximo capítulo serão mencionados os quatro tipos de Nuvem existentes e onde esses serviços podem ser encontrados.

2.5. NUVEM PRIVADA, PÚBLICA, HÍBRIDA E COMUNITÁRIA

Existem quatro tipos de Nuvem que podem ser utilizados e a seleção dependerá do tamanho da empresa, modelo de negócio, demandas, o que o consumidor espera e o tipo de aplicação/serviço que serão consumidos. São eles: Nuvem privada, pública, híbrida e comunitária.

Mell e Grande (2011, p. 3) definem cada Nuvem:

2.5.1. Nuvem Privada

A infraestrutura em Nuvem é fornecida para uso exclusivo de uma única organização compreendendo vários consumidores. Ela pode ser de propriedade, gerenciado e operado pela organização, ou um terceiro ou até mesmo uma combinação dos dois.

2.5.2. Nuvem Pública

A infraestrutura em Nuvem é fornecida para uso do público em geral. Pode ser pertencente, administrado e operado por uma organização empresarial, acadêmica ou governamental, ou alguma combinação deles.

2.5.3. Nuvem Híbrida

Esse modelo é composto de duas ou mais nuvens distintas, que pode ser a privada e a pública, por exemplo. Permanecem entidades únicas, mas estão vinculadas juntos por tecnologia padronizada ou proprietária que permite dados e aplicativos.

2.5.4. Nuvem Comunitária

É uma infraestrutura fornecida para uso exclusivo de uma comunidade de consumidores de organizações que compartilham preocupações (por exemplo, missão, requisitos de segurança, política e considerações de conformidade). Pode ser de propriedade, administrado e operado por uma ou mais organizações da comunidade, ou um terceiro, ou alguma combinação deles, e pode existir dentro ou fora das instalações.

2.6. O QUE DEVE SER CONSIDERADO PARA ADOÇÃO DA NUVEM

A jornada da transformação de cada empresa ou pessoa é única, a estratégia de adoção de Nuvem também deve ser exclusiva.

Os pontos mais importantes a serem considerados, segundo COGNITIVE CLASS (2021), no momento da decisão para a mudança ou a migração para a nuvem são agilidade, flexibilidade e competitividade. Não pode haver interrupções nos negócios ou problemas relacionados a segurança, conformidade e desempenho.

COGNITIVE CLASS (2021) descreve os pontos importantes a serem considerados para adoção da nuvem:

- Infraestrutura e carga de trabalho. Os custos na construção e operação de *Data Centers* podem ser elevados. Por outro lado, com a nuvem, os baixos custos iniciais e o pagamento conforme o uso podem trazer economia.
- Deve-se verificar se as cargas de trabalho estão prontas para a nuvem.
- É necessário considerar se é mais vantajoso pagar pelo acesso a um aplicativo ou comprar o *software* e posteriormente pagar pelas atualizações.
- Considerar a velocidade e a produtividade.
- Comparar os gastos em horas de quando um aplicativo é executado na nuvem *versus* algumas semanas, ou meses em plataformas tradicionais.
- Deve ser considerado o impacto de tomar uma má decisão. Qual o risco? Por exemplo, é mais arriscado investir ou alugar o *hardware* ou *software*? É melhor trabalhar em um plano de doze meses para construir, testar e liberar uma aplicação? Ou é melhor testar algo novo e pagar conforme o uso?

2.7. BENEFÍCIOS x RISCO DA ADOÇÃO DA NUVEM

A nuvem traz flexibilidade, eficiência e valor estratégico. Segundo COGNITIVE CLASS (2021) entre os benefícios pode-se destacar:

- Os usuários podem aumentar ou reduzir os serviços conforme necessidade, personalizar aplicativos, conectar de qualquer dispositivo que tenha conexão com a Internet;
- A infraestrutura da nuvem é dimensionada sob demanda para dar suporte a variações nas cargas de trabalho;
- As organizações podem determinar os níveis de controle e acesso. Em nuvens privadas é possível usar criptografia e chaves de API para ajudar a manter os dados seguros;
- A nuvem traz eficiência: é possível subir uma aplicação rapidamente sem se preocupar com custos de infraestrutura e manutenção;
- As falhas de *hardware* não resultam em perda de dados devido a *backups* em rede;

- Utiliza-se recursos remotos: economiza às organizações o custo dos servidores e outros equipamentos;
- Traz vantagem competitiva fornecendo as soluções mais inovadoras permitindo que as organizações foquem em outras prioridades;

Entre os desafios na adoção da nuvem ainda segundo COGNITIVE CLASS (2021) estão:

- Segurança dos dados, associados a perda ou indisponibilidade que causam interrupções nos negócios;
- Questões de governança, legais, regulatórias e de conformidade. Falta de padronização na forma como as tecnologias em evolução sem integram;
- Preocupações relacionadas à continuidade do negócio e recuperação de desastres;
- Parcerias com o provedor de nuvem que não atendem a todas as demandas do cliente;

A nuvem permite que as empresas experimentem, falhem e aprendam muito mais rápido com uma exposição de baixo risco.

De acordo com um estudo da IBM *Institute for Business Value* ([s.d.]), mais de três quartos das empresas estão usando a computação em nuvem para expansão dos seus negócios; 74% adotaram a nuvem para melhorar a experiência do cliente; e 71% usam a nuvem para criar produtos e serviços, enquanto simultaneamente reduz o tamanho dos sistemas legados e reduz os custos (apud COGNITIVE CLASS, 2021).

Para se manterem competitivas, as empresas precisam ser capazes de responder rapidamente as mudanças de mercado, realizar análises para entender a experiência do cliente e adaptar seus produtos e serviços com base no que aprendem.

Nesta nova era, tecnologias como IoT: Internet das Coisas, *Big Data*, *Business Intelligence*, Inteligência Artificial e *Blockchain* estão mudando os modelos de negócios existentes e indústrias, criando oportunidades sem precedentes para as empresas se diferenciarem e criarem valor para os seus clientes. O poder, a escalabilidade, natureza dinâmica e a economia dos recursos fornecem a base para a transformação e tornam a computação em nuvem a solução ideal para a adoção e evolução dessas tecnologias emergentes.

2.8. NATIVO DE NUVEM

Quando o cliente utiliza a nuvem para resolver desafios, como centro de dados, melhorar as experiências, um diferenciador comercial, esse sistema é conhecido como nativo de nuvem. Mas o fato de executar uma aplicação na nuvem não quer dizer que ela pode ser considerada uma “nativa de nuvem”, são necessárias algumas características específicas para determinar essa condição. Segundo Arundel e Domingus (2019, p. 40 e 41) as principais características de sistemas nativos de nuvem são:

2.8.1. Passível de automação

Se é esperado que as aplicações sejam implantadas e gerenciadas por máquinas, e não por seres humanos, elas devem obedecer a padrões, formatos e interfaces comuns.

2.8.2. Onipresente e flexível

Pelo fato de estarem desacopladas dos recursos físicos como os discos, ou de qualquer conhecimento específico sobre o nó de processamento em que, por acaso, estiverem executando, os microsserviços containerizados podem ser facilmente movidos de um nó para outro, ou até mesmo de um cluster para outro.

2.8.3. Resiliente e escalável

Aplicações tradicionais tendem a ter pontos únicos de falha: a aplicação deixará de funcionar se seu processo principal falhar, se a máquina subjacente tiver uma falha de *hardware* ou se um recurso de rede ficar congestionado. Aplicações nativas de nuvem, por serem essencialmente distribuídas, podem ter alta disponibilidade por meio de redundância.

2.8.4. Dinâmico

Um orquestrador é capaz de escalonar contêineres tirando o máximo proveito dos recursos disponíveis. Ele pode executar várias cópias deles a fim de proporcionar alta disponibilidade e realizar atualizações contínuas (*rolling, updates*) para um *upgrade* suave dos serviços, sem descartar tráfego.

2.8.5. Observável

Aplicações nativas de nuvem, por sua natureza, são mais difíceis de inspecionar e de depurar. Desse modo, um requisito essencial dos sistemas distribuídos é a observabilidade: monitoração, *logging*, *tracing* e métricas ajudam a entender o que os sistemas estão fazendo.

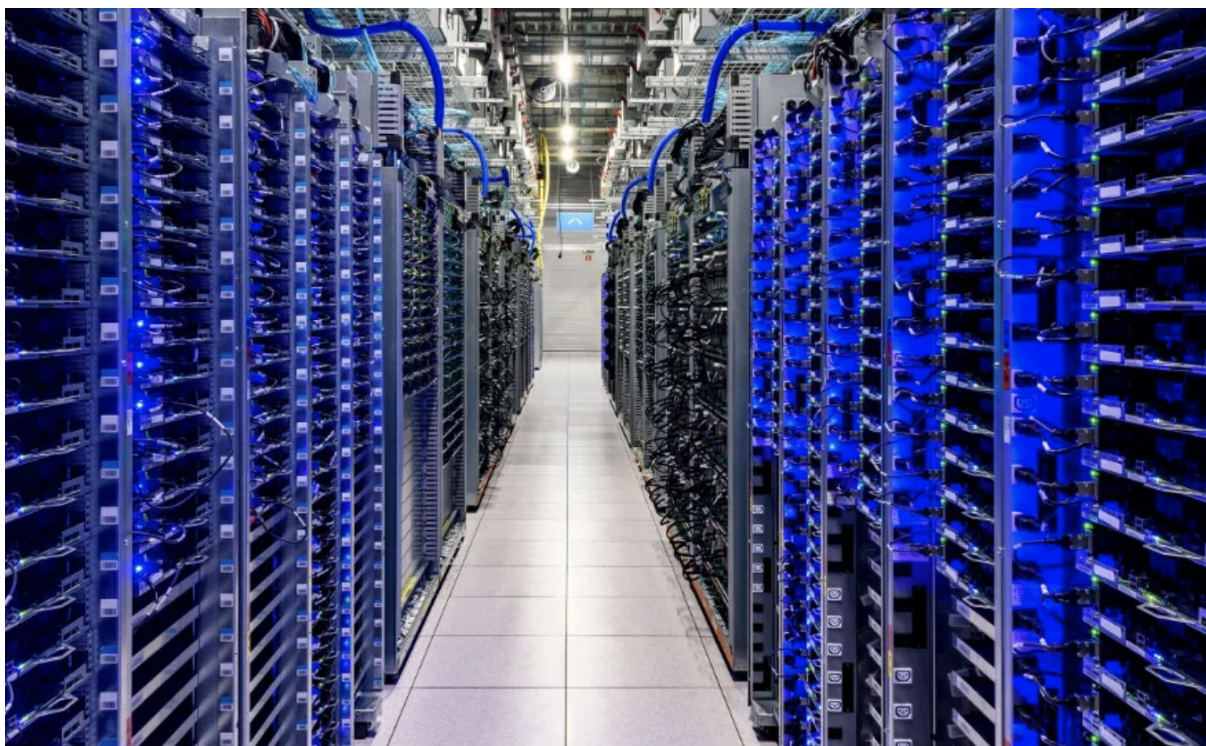
2.8.6. Distribuído

A abordagem nativa de nuvem é uma forma de construir e executar aplicações que tirem proveito da natureza distribuída e descentralizada da nuvem. Trata-se de como a aplicação funciona, e não do local que ela executa. Em vez de fazer a implementação de um código como uma única entidade (conhecida como monólito), aplicações nativas de nuvem tendem a ser compostas de vários microsserviços distribuídos, que cooperam entre si. Um microsserviço nada mais é do que um serviço autocontido que executa uma tarefa. Ao reunir os microsserviços terá uma aplicação. Assunto que será tratado mais adiante.

2.9. INFRAESTRUTURA DA NUVEM

A camada base da nuvem é a infraestrutura. Essa camada consiste em recursos físicos que estão localizados em Regiões, Zonas e *Data Centers* (figura 9).

Figura 9. *Data Center* no Chile



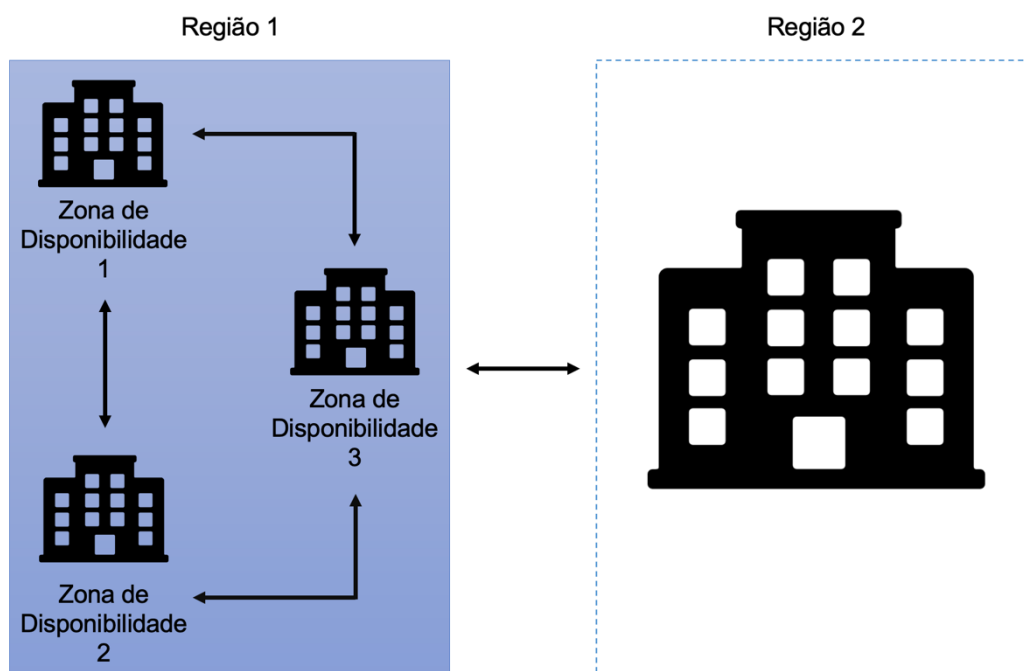
Fonte: Tele.Sintese, 2018

Segundo a COGNITIVE CLASS (2021) e apresentado na figura 10, a Região é definida como uma área geográfica ou local onde a infraestrutura de um provedor de nuvem é agrupado. As regiões são isoladas uma das outras. Se uma passar por um desastre, como por exemplo, um terremoto, as operações nas outras regiões continuam funcionando. Elas estão localizadas em diversas partes do mundo e cada uma delas podem ter de uma a várias Zonas de Disponibilidade ou simplesmente, Zonas que são *Data Centers* distintos com seus próprios recursos de energia, refrigeração e rede. Um *Data Center* é uma grande sala ou galpões que contém *pods*, *racks*, servidores, equipamentos de redes etc

O isolamento das Zonas melhora a tolerância geral a falhas, diminui a latência e evita a criação de pontos de falhas compartilhados.

Através da Internet, usando conectividade de rede com largura de banda muito alta as zonas e regiões são conectadas uma às outras.

Figura 100. Regiões



Fonte: adaptado de COGNITIVE CLASS, 2021

Como recursos os provedores de nuvem podem oferecer diversas opções de computação: Servidores Virtuais, Servidores *Bare Metal*, *Serverless*. Grande parte dos servidores em um *Data Center* executa *Hypervisor* para criar servidores virtuais ou mais conhecidas como máquinas virtuais, VMs. *Bare Metal* é o servidor físico e ainda existe a possibilidade de usar os recursos sem servidor que é uma camada de abstração sobre as máquinas virtuais, *Serverless*. VM e *Bare Metal* serão explicados mais detalhadamente nos próximos capítulos.

Quando os servidores são criados é necessário configurar sua interface de rede que pode ser pública ou privada. As interfaces de rede pública, como o nome sugere, conectam os servidores à rede pública Internet, enquanto os privados fornecem conectividade com seus próprios recursos de nuvem e ajuda mantê-los seguros.

Assim como no mundo físico de tecnologia, as interfaces de rede na nuvem precisam ter endereços IP e sub-redes que podem ser atribuídos de maneira automática ou configurados manualmente. Também é importante configurar qual tráfego de rede e usuários podem ter acesso aos recursos podendo ser utilizados Grupos de Segurança e Lista de Controle de Acesso (ou ACLs). Para maior segurança e isolamento de seus recursos na nuvem, a maioria dos provedores de nuvem

forneem Redes Locais Virtuais (VLANs), Nuvens Privadas Virtuais (VPCs) e Redes Privadas (VPNs). O tema Segurança ser tratado mais adiante.

Alguns dos dispositivos de *hardware* tradicionais, como *firewalls*, balanceadores de carga, *gateways* e analisadores de trfego t tambm podem ser virtualizados e disponibilizados como servios na nuvem.

Outro recurso de rede fornecido pelos provedores de nuvem so as redes de entrega de contedo ou CDNs, que distribuem contedo para vrios pontos em todo o mundo para que os usurios acessem.

Na infraestrutura, tambm, so oferecidas opoes de armazenamento chamados de *Local Storage*, *Block Storage*, *File Storage* e *Object Storage*, esse ltimo  o mais usado. Mais adiante nesse trabalho ser detalhado sobre cada armazenamento.

Os provedores de Nuvem possuem servios tradicionais como rotas, *switches* e rede definida por *software* (ou SDN) opoes onde certos recursos de rede so virtualizados ou disponibilizado por meio de APIs. Isso permite provisionamento, configurao e um fcil gerenciamento da rede.

A infraestrutura em nuvem est constantemente avanando e sendo aprimorada.

2.9.1. VIRTUALIZAO, MQUINAS VIRTUAIS

Segundo definio de Scarfing, Souppaya, Hoffman (2011, p. ES-1), a virtualizao  a simulao do *software* e/ou *hardware* no qual outro *software*  executado. Esse ambiente simulado  chamado de mquina virtual (VM). Existem muitas formas de virtualizao, distinguidos principalmente pela camada de arquitetura de computao. O que torna a virtualizao vivel  o chamado *Hypervisor* que  a parte de um *software* que  executado acima do servidor fsico ou *host*. Existem dois tipos principais de *Hypervisor*:

2.9.1.1. Tipo 1

Instalado diretamente na parte superior do servidor fsico, tambm chamado *Hypervisor Bare-Metal*. Eles so considerados mais seguros, reduzem a latncia ou atraso. Exemplos: *VMware*, *ESXi* ou *Microsoft Hyper-v*, open-source KVM, etc

2.9.1.2. Tipo 2

Possui uma camada de sistema operacional *host* que fica entre o servidor físico e o *Hypervisor*. Por essa natureza, eles também são chamados de *Hosted*. São mais usados por usuário final que deseja realizar a virtualização na máquina de uso pessoal e quase não são usados na nuvem. Exemplos: Oracle, VirtualBox e estação de trabalho.

Outra definição para VM, dada pela COGNITIVE CLASS (2021), é que se trata simplesmente de um computador baseado em *software*. São executadas como um computador físico, possuem um sistema operacional e aplicativos que são completamente independentes um do outro. Pode-se executar vários aplicativos em um *Hypervisor*, ele gerencia os recursos que são alocados para esses ambientes virtuais do servidor físico, são independentes e pode-se executar diferentes sistemas operacionais em diferentes máquinas. Por exemplo, o Windows pode ser executado ao mesmo tempo que o Linux e o Unix.

Por serem independentes, também são extremamente portáteis. Pode-se mover de maneira quase instantânea uma máquina virtual de um *Hypervisor* para outro, o que lhe dá flexibilidade e portabilidade dentro do seu ambiente. O fato de executar vários ambientes ou aplicações em uma parte da infraestrutura, significa uma grande redução no uso de infraestrutura física. O que traz economia nos custos de manutenção, gera agilidade e velocidade. Se um dos *hosts* cair pode-se simplesmente mover as VMs, de maneira rápida, para outro *Hypervisor* que está funcionando. Virtualização e VMs estão no centro da nuvem.

2.9.2. BARE METAL

Definição dada pela COGNITIVE CLASS (2021), um servidor *Bare Metal* é um *Single-Tenant*, ou seja, dedicado a um único cliente. O provedor de nuvem conecta um servidor físico a um *rack* em um *Data Center* para uso do cliente. Ele gerencia o servidor até o sistema operacional (SO), o que significa que se algo der errado com o *hardware* ou a conexão do *rack*, eles consertarão ou substituirão e reiniciarão o servidor. O cliente responsável por administrar e gerenciar todo o resto no servidor.

O servidor *Bare Metal* é pré-configurado pelo provedor de nuvem para atender aos pacotes de carga de trabalho ou também podem ser configurados de acordo com as especificações do cliente, isso inclui processadores, RAM, discos rígidos,

componentes especializados e o sistema operacional. O cliente também pode instalar seus próprios sistemas operacionais e pode instalar determinados *Hypervisor* que não estão disponíveis no provedor de nuvem, e, assim, criar suas próprias máquinas virtuais.

Como os servidores *Bare Metal* são máquinas físicas, eles demoram mais para serem configurados do que os servidores virtuais. Instalações pré-configuradas podem levar de 20 a 40 minutos para serem habilitadas/carregadas e as compilações personalizadas podem levar cerca de três ou quatro horas. Esses prazos para configuração podem variar conforme o provedor.

Como não há compartilhamento de *hardware* de servidor subjacente com outros clientes, os servidores *Bare Metal* atendem as exigentes necessidades de computação de alto desempenho (HPC) pode-se adicionar GPUs (*Graphics Processing Unit* ou Unidade de Processamento Gráfico), que são projetadas para acelerar a computação científica, análise de dados e renderização de gráficos virtualizados de nível profissional. Alguns exemplos de carga de trabalho que eles atendem são ERP, CRM, Inteligência Artificial e *Deep Learning*. Aplicativos de dados intensos que exigem atrasos mínimos relacionados à latência.

O servidor é dedicado e destinado a uso de longo prazo, possui alto desempenho em ambientes altamente seguros e isolados. Podem ser ampliados e otimizados para alta disponibilidade, conforme necessário.

Para casos de aplicativos que exigem altos níveis de controle de segurança ou aplicativos que normalmente executado em um ambiente local, um servidor *Bare Metal* é uma boa alternativa na nuvem.

2.9.3. BARE METAL X MÁQUINAS VIRTUAIS

Ao comparar servidores *Bare Metal* com servidores virtuais ou máquinas virtuais, comparação feita pela COGNITIVE CLASS (2021) eles funcionam melhor para cargas de trabalho intensivas, se destacam com o mais alto desempenho e segurança, atendem a requisitos rigorosos de conformidade, oferecem total flexibilidade, controle e transparência.

Já os servidores virtuais são provisionados rapidamente, fornecem uma solução elástica e escalável, são de baixo custo, no entanto, uma vez que compartilham *hardware* subjacente com outros servidores virtuais, eles podem ser limitados em taxa de transferência e desempenho.

O *Bare Metal* tende a ser mais caros e nem todos os provedores de nuvem fornecem esse tipo de servidor.

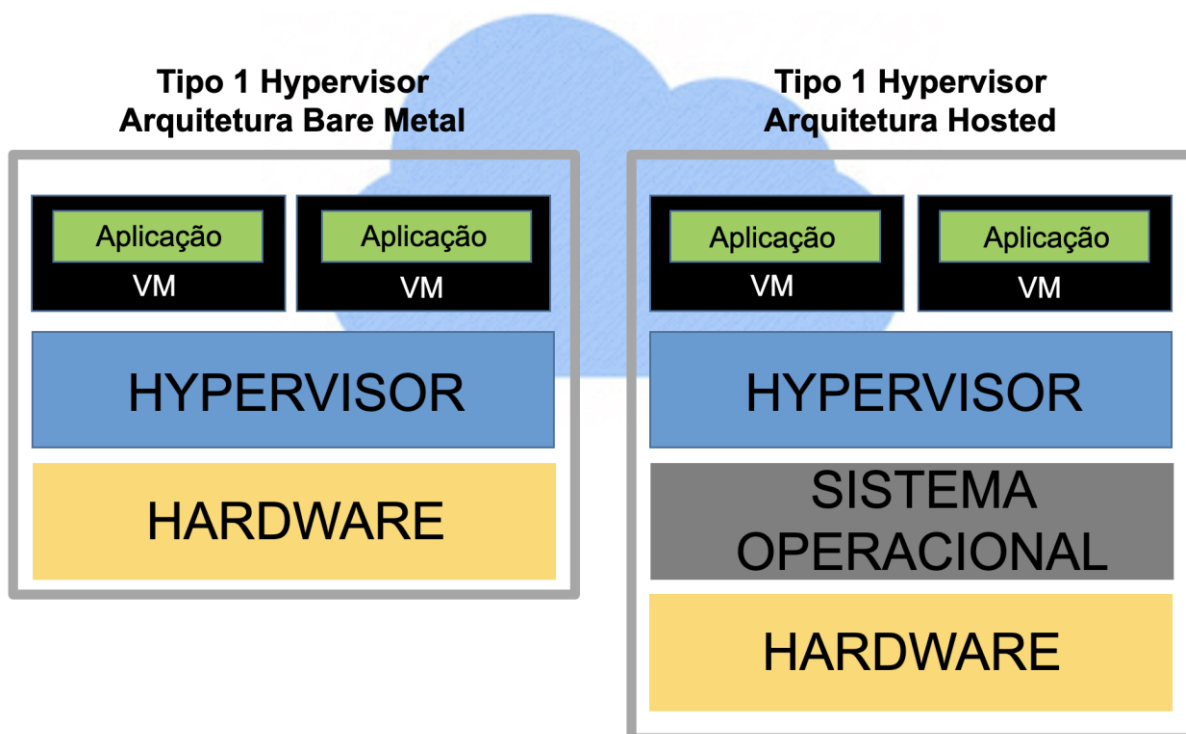
2.9.4. CONTÊINERES

Após a explicação sobre máquinas virtuais é importante trazer um capítulo dedicado aos contêineres que conforme definição do NETAPP (2022) empresa americana, eles também são uma forma de virtualização do sistema operacional. Um único contêiner pode ser usado para executar qualquer coisa, desde um pequeno microserviço ou processo de *software* até um aplicativo maior. Dentro de um contêiner estão todos os executáveis necessários, código binário, bibliotecas e arquivos de configuração. Ele usa um sistema de arquivos isolados, as chamadas imagens que contêm outras configurações como variáveis de ambiente, comando padrão a ser executado, e outros metadados. É um pacote padronizado.

Em implantações de aplicativos maiores, vários contêineres podem ser implantados com um ou mais *clusters* de contêiner. Esses *clusters* podem ser gerenciados por um orquestrador de contêiner, como o Kubernetes, que será detalhado em capítulo futuro.

É possível confundir a tecnologia de contêiner com máquinas virtuais (VMs) ou tecnologia de virtualização de servidor. Embora existam algumas semelhanças básicas, os contêineres são muito diferentes das VMs (figura 11).

As máquinas virtuais são executadas em um ambiente de *hypervisor* em que cada máquina virtual deve, como já mencionado, incluir seu próprio sistema operacional, juntamente com seus binários, bibliotecas e arquivos de aplicativos relacionados. Isso consome uma grande quantidade de recursos e sobrecarga do sistema, especialmente quando várias VMs estão sendo executadas no mesmo servidor físico, cada uma com seu próprio sistema operacional. Em contraste, cada contêiner compartilha o mesmo sistema operacional *host* ou *kernel* do sistema e é muito mais leve em tamanho, geralmente apenas alguns *megabytes* e portáteis.

Figura 11. *Bare Metal X Hosted*

Fonte: adaptado de COGNITIVE CLASS, 2021

2.9.4.1. BENEFÍCIOS DO CONTÊINER

Os contêineres exigem menos recursos do sistema do que os ambientes de máquina virtual tradicional ou de *hardware* porque não incluem imagens do sistema operacional. Existe uma menor sobrecarga do sistema operacional. Segundo a NETAPP (2022) pode ser citados os seguintes benefícios:

- Maior portabilidade: os aplicativos executados em contêineres podem ser implantados facilmente em vários sistemas operacionais e plataformas diferentes. Segundo a Microsoft (2022) “Embora os contêineres sejam portáteis, são restritos ao sistema operacional para o qual foram definidos. Por exemplo, um contêiner do Linux não pode ser executado no Windows e vice-versa.”
- Maior eficiência: permitem que os aplicativos sejam implantados, corrigidos ou dimensionados mais rapidamente.
- Melhor desenvolvimento de aplicativos: suportam esforços ágeis e DevOps para acelerar os ciclos de desenvolvimento, teste e produção.

2.9.5. DOCKER, KUBERNETES E OS CONTÊINERES

Considerando o cenário de contêineres é importante falar sobre o Docker que foi desenvolvido em *open source*, para ser utilizado dentro dos ambientes operacionais mais comuns, Linux, Microsoft Windows e outras infraestruturas locais ou baseadas em nuvem.

A documentação sobre o Docker (2021) define que ele é um ambiente de tempo de execução usado para criar e construir *software* dentro de contêineres. Os contêineres usam imagens do Docker para implantar aplicativos ou *software* em vários ambientes, desenvolvimento, teste e produção.

Os aplicativos em contêiner podem exigir centenas a milhares de contêineres separados, principalmente no ambiente de produção. Quando existe um alto volume de contêineres é necessário usar um gerenciador, ou mais conhecido como orquestrador de contêineres. Uma das ferramentas mais populares para essa finalidade é o Kubernetes.

2.9.6. KUBERNETES

Kubernetes ou K8s é um orquestrador de contêineres criado em 2014 pelo Google. A idéia era desenvolver um sistema que todos pudessem usar, *open source*, com base em lições aprendidas com o Borg. Esse é um sistema de gerenciamento centralizado que aloca e escala contêineres para executar em um conjunto de servidores e, também, foi criado pelo Google.

No final de 2017, Kubernetes era o orquestrador mais conhecido e utilizado no mercado segundo Arundel e Domingus (2019, p. 36). A explicação para isso foi dada por Hightower (apud ARUNDEL; DOMINGUS, 2019, p. 36):

O Kubernetes faz as tarefas que o melhor dos administradores de sistemas faria: automação, failover, logging centralizado e monitoração. Ele toma o que aprendemos com a comunidade de DevOps e transforma esse conhecimento em algo pronto para uso imediato.

Muitas das tarefas tradicionais de um administrador de sistemas como *upgrade* de servidores, instalação de correções de segurança, configuração de rede e execução de *backups* são menos preocupantes no mundo nativo de nuvem. O Kubernetes é capaz de automatizar essas tarefas.

Alguns dos recursos como balanceamento de carga e escalabilidade automática, estão incluídos no núcleo de Kubernetes; outros são disponibilizados por

add-ons, extensões e ferramentas de terceiros que usam a API do Kubernetes. Arundel e Domingus (2019)

Kubernetes foi adicionado a esse trabalho final de conclusão de curso para realização de demonstrações práticas de várias características da nuvem. No próximo tópico, serão detalhados conceitos importantes sobre essa ferramenta.

2.9.6.1. ARQUITETURA KUBERNETES

Para fazer uso do *Kubernetes* é necessário conhecer sua arquitetura básica que, conforme figura 12, é composta pelo *Cluster* que contém o *Node* que por sua vez contém os *Pods* e a *Master* que realiza a orquestração dos contêineres. GEEK UNIVERSITY (2022) define cada componente que serão mencionados a seguir.

2.9.6.1.1. Pod

Menor unidade de computação publicável em um computador que pode ser criada e gerenciada, dentro dele está o contêiner com uma imagem que pode ser de uma aplicação, por exemplo, um banco de dados. Por default, um node aceita até 110 pods, mas essa quantidade pode ser alterada.

2.9.6.1.2. Node

Máquina, física ou virtual, onde o Kubernetes está instalado, nele é criado os contêineres com as aplicações.

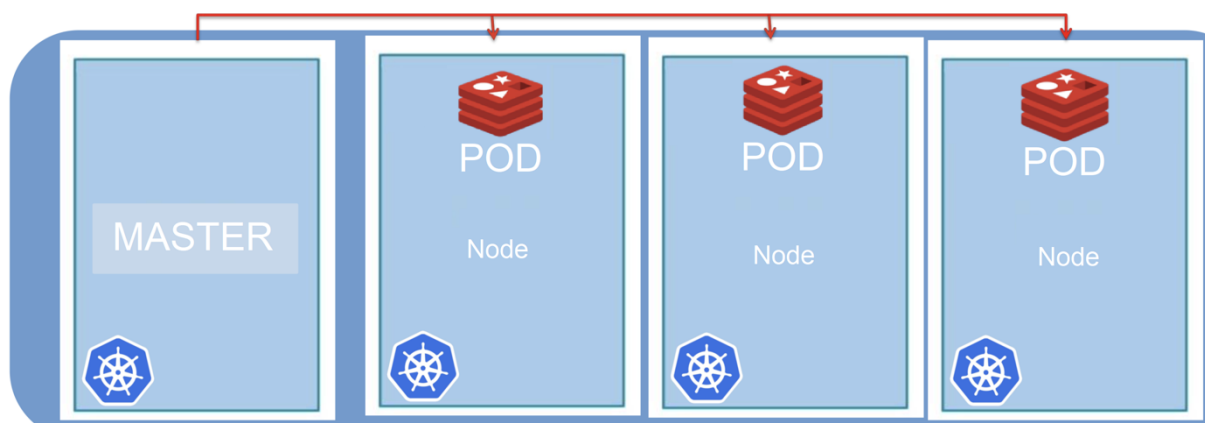
2.9.6.1.3. Cluster

Conjunto de nodes agrupados.

2.9.6.1.4. Master

Responsável por gerenciar, manter as informações do Cluster, monitorar os nodes e fazer algo, por exemplo, criar ou não um novo node quando ele falhar.

Figura 12. Funcionamento do Kubernetes



Fonte: adaptado de GEEK UNIVERSITY, 2022

Segundo a THE KUBERNETES AUTHORS (2022) um *cluster* precisa ser:

2.9.6.1.5. Seguro

Seguir as práticas recomendadas de segurança mais avançadas.

2.9.6.1.6. Fácil de usar

Alguns comandos são suficientes para operá-lo.

2.9.6.1.7. Extensível

Ele não deve favorecer um fornecedor e precisa ser personalizável por meio de um arquivo de configuração.

2.9.6.2. COMPONENTES

Quando o Kubernetes é instalado vários componentes são instalados em conjunto. Conforme encontrado na documentação do OS AUTORES DO KUBERNETES (2022) nos próximos subcapítulos estão as definições e utilização dos principais.

2.9.6.2.1. API Server

Painel de controle do Kubernetes.

2.9.6.2.2. Etcd

Usado para armazenamento no formato chave-valor.

2.9.6.2.3. Kubelet

Garante que os contêineres estejam sendo executados em um *pod*.

2.9.6.2.4. Container Runtime

Software responsável pela execução dos contêineres.

2.9.6.2.5. Control Plane

Tomam decisões globais sobre o cluster (por exemplo, agendamento), além de detectar e responder a eventos do cluster como, por exemplo, iniciar um novo *pod* quando o campo de réplicas de uma implantação apresentar falhas.

2.9.6.2.6. Scheduler

Observa os pods recém-criados sem nenhum node atribuído e seleciona um node no qual eles serão executados. Os fatores levados em consideração para as decisões incluem: requisitos de recursos individuais e coletivos, restrições de hardware/software/políticas, especificações de afinidade, localidade dos dados, interferência entre cargas de trabalho e prazos.

2.9.6.2.7. Namespaces

Mecanismo para isolar grupos de recursos dentro de um único *cluster*.

2.9.6.2.8. NodePort

Expõe o serviço sobre a mesma porta em cada nó selecionado no *cluster* utilizando NAT. Faz o serviço acessível externamente.

2.9.6.2.9. ClusterIP

Expõe o serviço sob um endereço IP interno no *cluster*. Este tipo faz do serviço somente alcançável de dentro do *cluster*.

2.9.7. MICROSSERVIÇOS

Como já mencionado nesse trabalho na nuvem existe a possibilidade da escalabilidade. Existem dois tipos: a escalabilidade vertical que é o aumento da memória *RAM* ou o espaço no *HD* e a escalabilidade horizontal que é quando uma máquina ou serviço é replicado N vezes.

Segundo o instrutor da GEEK UNIVERSITY (2022), a escalabilidade horizontal, que pode ser facilmente implementada na nuvem, teve um aumento no seu uso com o crescente número de APIs e rotas cada vez mais genéticas. Com isso as empresas começaram a dividir a lógica de negócios em pequenas partes independentes que se completam, criando uma espécie de rede de APIs internas totalmente ou parcialmente conectadas. Os chamados microsserviços.

A SMARTBEAR SOFTWARE (2022) define que microsserviço é um método de desenvolvimento de sistemas de *software* que tenta se concentrar na construção de módulos de função única com *interfaces* e operações bem definidas. Na forma mais simples, os microsserviços ajudam a construir um aplicativo como um conjunto de pequenos serviços, cada um executando em seu próprio processo e implementáveis de forma independente. Esses serviços podem ser escritos em diferentes linguagens de programação e podem usar diferentes técnicas de armazenamento de dados. O que resulta no desenvolvimento de sistemas escaláveis e flexíveis.

Segundo Fowler ([s.d.]), “microsserviço é uma abordagem que desenvolve um aplicativo único como uma suíte de pequenos serviços.” (apud GEEK UNIVERSITY, 2022).

2.9.7.1. CINCO CARACTERÍSTICAS DOS MICROSSERVIÇOS

SMARTBEAR SOFTWARE (2022) define cinco características que podem ser encontradas nos microsserviços:

1. **Vários componentes:** O *software* construído com microsserviço pode, por definição, ser dividido em vários serviços de componentes para que cada um desses serviços possa ser implantado, ajustado e reimplantado de forma independente, sem comprometer a integridade de um aplicativo. Como resultado, talvez seja necessário alterar apenas um ou mais serviços distintos, em vez de reimplantar aplicativos inteiros.

2. **Construído para negócios:** O estilo de microsserviços geralmente é organizado em torno de recursos e prioridades de negócios. Ao contrário de uma abordagem de desenvolvimento monolítica tradicional – onde equipes diferentes têm um foco específico em, digamos, UIs, bancos de dados, camadas de tecnologia ou lógica do lado do servidor – a arquitetura de microsserviços utiliza equipes multifuncionais. As responsabilidades de cada equipe são fazer produtos específicos baseados em um ou mais serviços individuais que se comunicam via barramento de mensagens. Nos microsserviços, uma equipe possui o produto por toda a vida.

3. **Roteamento simples:** Os microsserviços agem um pouco como o sistema UNIX clássico: eles recebem solicitações, processam-nas e geram uma resposta de acordo. Pode-se dizer que os microsserviços têm *endpoints* inteligentes que processam informações e aplicam lógica, e canais considerado “burros” pelos quais as informações fluem.

4. **Descentralizado:** Como os microsserviços envolvem uma variedade de tecnologias e plataformas, a governança descentralizada é favorecida pela comunidade de microsserviços porque seus desenvolvedores se esforçam para produzir ferramentas úteis que podem ser usadas por outros para resolver os mesmos problemas. Assim como a governança descentralizada, a arquitetura de microsserviços também favorece o gerenciamento descentralizado de dados. Em um aplicativo de microsserviço, cada serviço geralmente gerencia seu banco de dados exclusivo.

5. **Resistente a falhas:** Os microsserviços são projetados para lidar com falhas. Como vários serviços estão se comunicando, é bem possível que um serviço falhe, por um motivo ou outro. Nesses casos, o cliente deve configurar para que os serviços continuem funcionando enquanto corrige as falhas, isso pode ser feito, por exemplo, através das réplicas. Também é possível realizar o monitoramento para evitar falhas.

Microserviço é o último tópico tratado na parte de infraestrutura da Nuvem. Nos próximos capítulos serão tratados tópicos importantes que não estão relacionados à infraestrutura.

2.10. ARMAZENAMENTO

Segundo COGNITIVE CLASS (2021), na nuvem é possível armazenar dados e arquivos. Alguns armazenamentos devem ser anexados a um *computer node* (nó de computação), que é um ponto de conexão, antes do acesso ao dado ou arquivo, enquanto outros podem ser acessados diretamente pela Internet pública ou por uma conexão de rede privada.

Os provedores de nuvem hospedam, protegem, gerenciam e mantêm o armazenamento e infraestrutura para garantir que o cliente tenha acesso aos seus dados e arquivos quando precisar. Eles permitem que o cliente dimensione a capacidade de armazenamento conforme necessário. O custo para esse serviço varia de acordo com o tipo de armazenamento selecionado, mas, em geral, quanto mais rápida for a velocidade de leitura/gravação, maior o custo por *gigabyte*.

Quando se trata de armazenamento pode-se ouvir os seguintes termos: IOPS (*Input/Output Operations Per Second*): Operações de entrada/saída por segundo, refere-se à velocidade de armazenamento ou a rapidez com que os dados podem ser lidos ou gravados (ABRAHOSTING - ASSOCIADO, 2022). *Persistence* (ou persistência) é um termo que define o que acontece com o armazenamento quando o nó de computação ao qual ele está conectado é encerrado. Se o armazenamento estiver definido como *persistence*, ele não será excluído junto com *computer node*, o que significa que ele será preservado e disponível para montagem em outro *computer node*. Também, em alguns casos, é possível definir o armazenamento para que seja excluído automaticamente com o *compute node* quando esse for excluído. O armazenamento em nuvem fornece maior fluxo de trabalho e segurança.

Os principais tipos de armazenamento são: *Local Storage* ou *Direct Attached*, *File Storage*, *Block Storage* e *Object Storage*. As definições do funcionamento de cada um serão mencionadas nos próximos parágrafos e são dadas pela IBM CLOUD EDUCATION (2019) e IBM (2022).

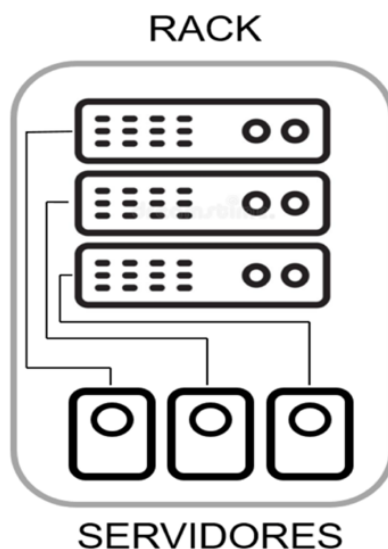
2.10.1. DIRECT ATTACHED

Direct Attached, também chamado de *Local Storage*, é o armazenamento em um servidor baseado em nuvem utilizando *digital storage system* conectado diretamente a um ou mais computadores sem uso de rede entre eles, representação na figura 13.

Esse armazenamento é rápido e normalmente usado apenas para armazenar o sistema operacional de um servidor, embora ele possa ter outros casos de uso.

As duas principais razões pelas quais o armazenamento *Direct Attached* não é tão bom para outros usos além de armazenar o sistema operacional é que normalmente ele é *Ephemeral*, que significa que dura apenas o tempo no recurso de computação ao qual está anexado, ele não pode ser compartilhado com outros nós. Embora possa ser usada as técnicas de RAID (*Redundant Array of Independent Disks* ou Conjunto Redundante de Discos Independentes), ele não é tão resiliente a falhas como outros tipos de armazenamento.

Figura 13. Armazenamento Direct Attached



Fonte: adaptado de COGNITIVE CLASS, 2021

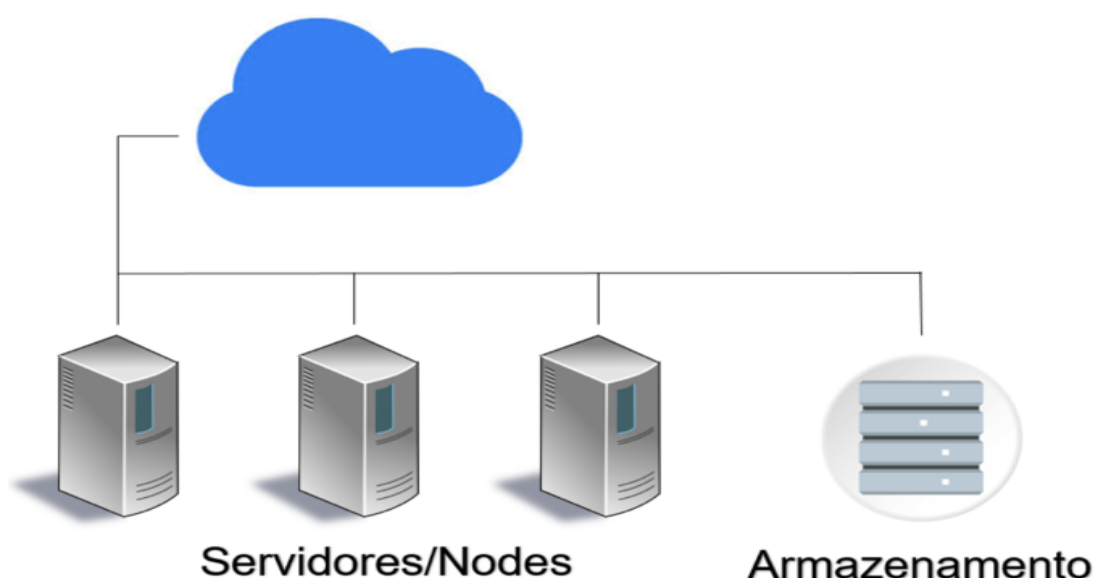
2.10.2. FILE STORAGE

É um armazenamento de arquivos com base em NFS (*Network File System*) baseado em *flash*, durável, rápido e flexível com IOPs personalizáveis e faturamento previsível.

O *File Storage* é uma abordagem simples e direta para armazenamento de dados e funciona bem para organização de dados em uma estrutura de pastas hierárquica, com a qual os usuários de *desktop* estão familiarizados.

O armazenamento de arquivos é normalmente apresentado aos nós de computação como NFS, significa que o armazenamento está conectado através de uma rede Ethernet padrão, que apesar de ser o mais comum, faz com que ele seja mais lento do que o *Direct Attached* ou o *Block Storage*. Uma vantagem do *File Storage* é que ele pode ser usado em vários servidores ao mesmo tempo (ver figura 14).

Figura 14. Armazenamento File Storage



Fonte: adaptado de COGNITIVE CLASS, 2021

2.10.3. BLOCK STORAGE

O *Block Storage*, às vezes chamado de armazenamento em nível de bloco, é uma tecnologia usada para armazenar arquivos de dados em redes de área de armazenamento (SANs) ou em ambientes de armazenamento com base em *Cloud*. Ele divide os dados em blocos e, em seguida, armazena esses blocos como peças separadas, cada uma com um identificador exclusivo. A SAN (*Storage Area Network*) coloca esses blocos de dados onde eles são mais eficientes. Isso significa que ele

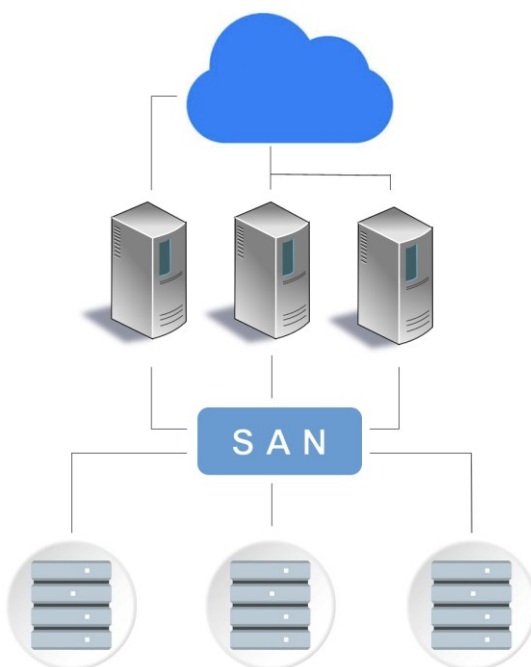
pode armazenar esses blocos em sistemas diferentes e cada bloco pode ser configurado (ou particionado) para funcionar em diferentes sistemas operacionais.

O armazenamento em bloco também separa os dados dos ambientes do usuário, permitindo que sejam distribuídos entre diversos ambientes conforme apresentado na figura 14. Isso cria vários caminhos para os dados e permite que o cliente os recupere rapidamente. Quando um usuário ou aplicativo solicita dados de um sistema de armazenamento em bloco, o sistema de armazenamento subjacente remonta os blocos de dados e apresenta os dados ao cliente ou aplicativo.

O *Block Storage* utiliza conexão com fibra, o que traz a velocidade para leitura e gravação. É mais confiável do que os outros tipos de armazenamento, o que o torna adequado para uso com bancos de dados e outros aplicativos em que a velocidade é importante.

O armazenamento é provisionado em blocos/volumes no disco rígido e eles só podem ser provisionados em um nó de computação por vez.

Figura 15. Armazenamento Block Storage



Fonte: adaptado de COGNITIVE CLASS, 2021

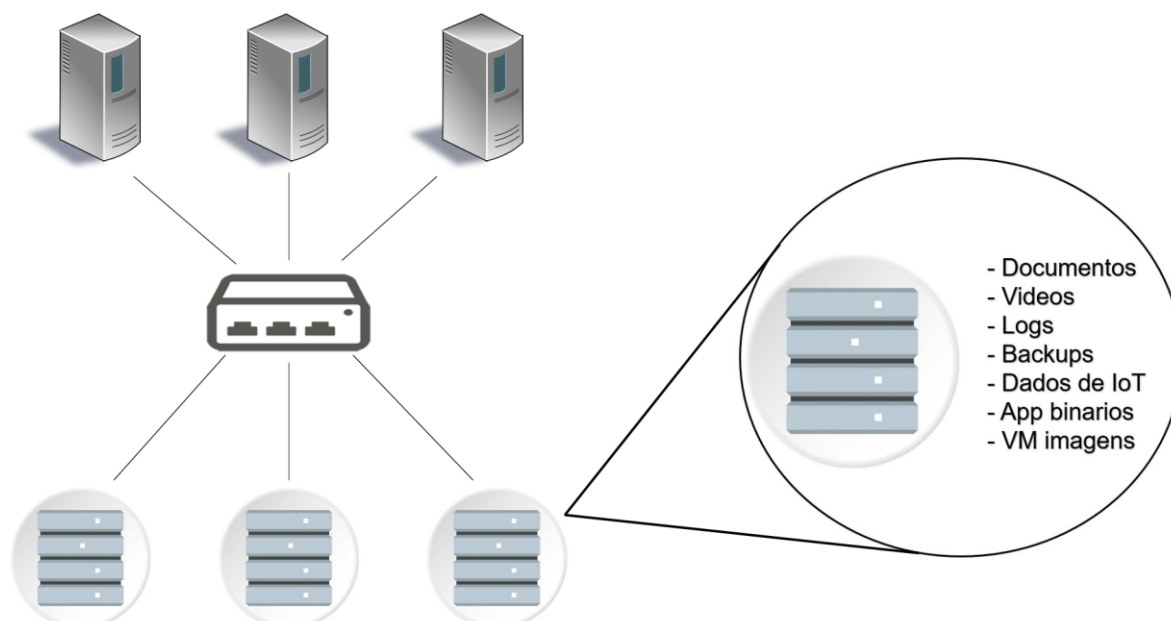
2.10.4. OBJECT STORAGE

Object Storage ou armazenamento de objeto é diferente dos armazenamentos de arquivo e de bloco porque gerencia dados como objetos. Cada objeto inclui os dados em um arquivo, além de seus metadados associados e de um identificador. Os objetos armazenam dados no formato em que eles chegam e permitem customizar os metadados a fim de tornar os dados mais fáceis de acessar e analisar. Em vez de organizá-los em hierarquias de arquivo ou pasta, os objetos são mantidos em repositórios que fornecem escalabilidade praticamente ilimitada (figura 16). Como não há uma hierarquia de arquivamento e os metadados são customizáveis, o armazenamento de objeto permite otimizar os recursos de armazenamento com uma boa relação custo-benefício.

De todos os tipos de armazenamento, o *Object Storage* é o mais barato e também o mais lento em termos de velocidade de leitura e gravação, mas possui tamanho infinito para armazenamento. Nele é possível adicionar uma quantidade significativa de dados. O que o torna um repositório fantástico para todos os tipos de dados não estruturados, grandes e pequenos, incluindo documentos, vídeos, *logs*, *backups*, dados de IoT, aplicativos binários, imagens de máquinas virtuais, etc.

Este é um tipo diferente de armazenamento, pois não está conectado a um nó de computação, em vez disso, é acessado por meio de uma API.

Figura 16. Armazenamento Object Storage



Fonte: adaptado de COGNITIVE CLASS, 2021

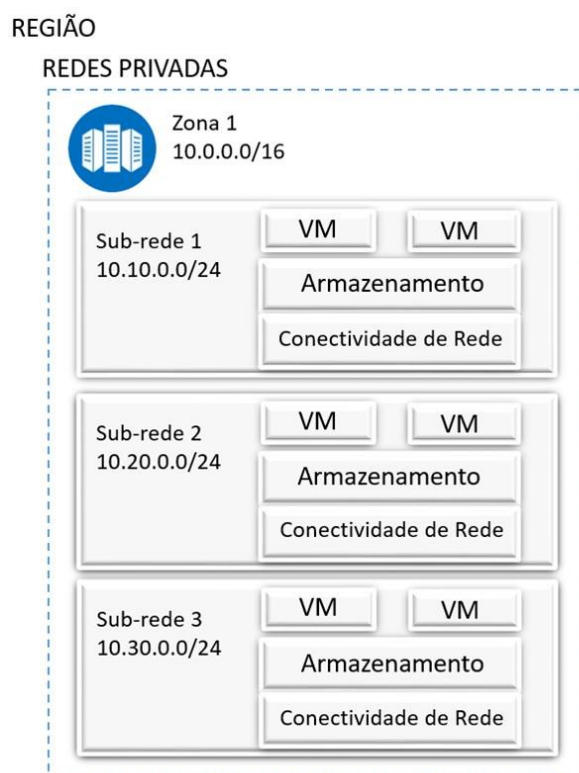
2.11. REDE

Segundo a COGNITIVE CLASS (2021), construir uma rede em nuvem não é muito diferente da implantação de uma rede em um *data center* local. A principal diferença decorre do fato de que, na nuvem, usa-se instâncias lógicas de elementos de rede em oposição a dispositivos físicos. Por exemplo, uma placa de *interface* de rede (NIC) seria representada por uma Interface de Rede Virtual ou vNIC. Na nuvem, as funções de rede são fornecidas como um serviço, e não na forma de montagem em um *rack* e/ou dispositivo.

Para criar uma rede na nuvem define-se o tamanho da rede, ou a faixa de endereços IP para estabelecer os limites. As redes em nuvem são implantadas em espaços de rede que são separados logicamente, que por sua vez podem ser divididos em segmentos menores chamados sub-redes.

Na figura 17 pode ser visto que recursos de nuvem, como por exemplo, VMs ou instâncias de servidor virtual, armazenamento, conectividade de rede, balanceadores de carga, etc são implantados em sub-redes.

Figura 17. Redes Privadas

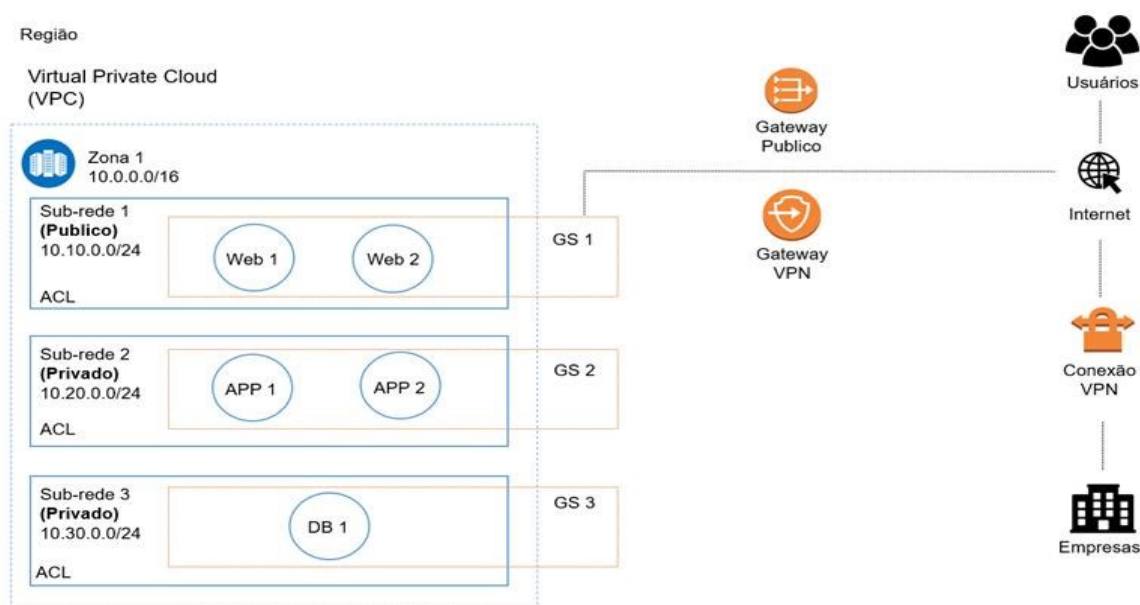


Fonte: adaptado de COGNITIVE CLASS, 2021

O uso de sub-redes permite que os usuários implantem aplicativos usando os mesmos conceitos usados em ambientes locais. As sub-redes também são a principal área onde a segurança é implementada na nuvem. Cada sub-rede é protegida por Listas de Controle de Acesso (ACLs) que servem como uma espécie de *firewall*. Dentro da sub-rede podem ser criados grupos de segurança.

Após a criação de uma sub-rede é possível adicionar instâncias para execução de aplicativos. Conforme figura 18 pode ser criado, por exemplo, um aplicativo com três camadas: uma que requer uma instância para acesso a *web*, outra para o aplicativo em si e a última para o banco de dados. Cada instância pode ser colocada em um grupo de segurança com suas próprias regras: com *gateway* público para permitir o acesso dos usuários ao aplicativo na camada de Internet e usar redes privadas (VPN).

Figura 18. Sub-rede e Aplicativos

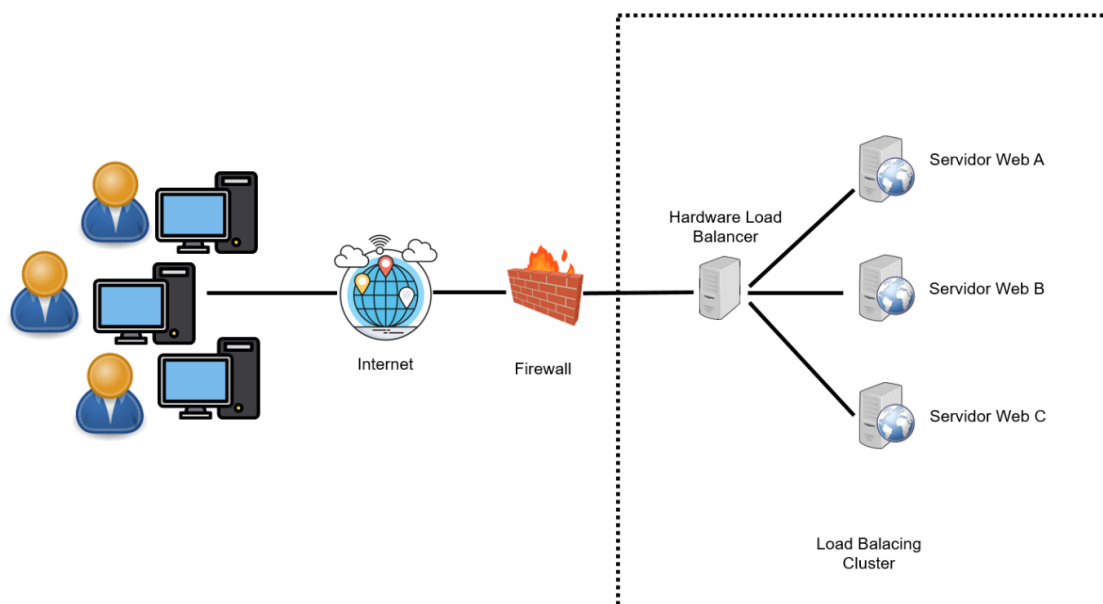


Fonte: adaptado de COGNITIVE CLASS, 2021

Ainda segundo COGNITIVE CLASS (2021), ao construir muitas sub-redes e implantar várias cargas de trabalho, torna-se necessário garantir que os aplicativos continuem a ser responsivos. O que é conseguido com balanceamento de carga ou o chamado *Load Balancer*, uma técnica que garante a disponibilidade de largura de banda para muitas requisições. Através dele é mantida a estabilidade de um servidor quando o tráfego ou o volume de dados é muito grande, porque ele faz a distribuição

de carga entre os diferentes servidores. Na figura 19, por exemplo, quando existe um acesso muito grande a um *site*, ele evita que ocorra uma sobrecarga e melhora o desempenho do sistema ao distribuir as requisições ao *site* para vários servidores.

Figura 19. Load Balancer



Fonte: adaptado de COGNITIVE CLASS, 2021

Construir uma rede em nuvem envolve a criação de um conjunto de construções lógicas que fornecem funcionalidade e garantem aplicativos de negócios de alto desempenho.

3. SEGURANÇA

Badger et al. (2012, p. 4-1) define: “[...] o termo computação em nuvem abrange uma variedade de sistemas e tecnologias, bem como modelos de serviço e implantação e modelos de negócios.”. Segundo o Instituto, os atributos exclusivos da computação em nuvem, como elasticidade, provisionamento, *pool* de recursos, multilocação, acessibilidade de rede ampla e ubiquidade, trazem muitos benefícios para os clientes, mas também envolvem riscos de segurança específicos associados ao tipo de nuvem adotada e ao modo de implantação. Para acelerar a adoção da computação em nuvem e avançar na implantação de serviços, as soluções que lidam com as ameaças à segurança devem ser abordadas. Muitas das ameaças que os provedores e clientes enfrentam podem ser tratadas por meio de processos e mecanismos de segurança tradicionais, como políticas de segurança, criptografia,

gerenciamento de identidade, sistemas de detecção/prevenção de intrusão e análise de vulnerabilidade. As atividades de gerenciamento de risco devem ser realizadas para determinar como diminuir as ameaças específicas de diferentes modelos de nuvem.

Proteger os sistemas de informação e garantir a confidencialidade, integridade e disponibilidade das informações que estão sendo processadas, armazenadas e transmitidas são de alta prioridade e apresentam um risco maior de serem comprometidas em um sistema de computação em nuvem.

As implementações na nuvem estão sujeitas a ameaças físicas locais, bem como ameaças externas remotas. Essas ameaças incluem acidentes, desastres naturais que induzem a perda externa de serviço, governos hostis, organizações criminosas, grupos terroristas e vulnerabilidades maliciosas ou não intencionais exploradas por meio de acesso interno, externo, autorizado ou não autorizado ao sistema.

Os possíveis desafios de segurança para serviços de computação em nuvem mencionados por Sill et al. (2013, p. 46):

- Compromissos com a confidencialidade e integridade dos dados;
- Ataques que aproveitam a uniformidade e o poder dos sistemas de computação em nuvem para escalar rapidamente e aumentar de tamanho;
- Acesso não autorizado;
- Aumento dos níveis de ataques baseados em rede que exploram *software* não projetado para um modelo baseado na Internet e vulnerabilidades existentes em recursos anteriormente acessados por meio de redes privadas;
- Restrições de portabilidade resultantes da falta de padronização das *interfaces* de programação de aplicativos (APIs) que impedem que os clientes migrem facilmente para um novo provedor quando os requisitos de disponibilidade não são atendidos;
- Ataques que exploram a falta de transparência nos procedimentos ou registros de auditoria;
- Ataques que aproveitam vulnerabilidades conhecidas e antigas em máquinas virtuais que não foram devidamente atualizadas e corrigidas;
- Ataques que exploram inconsistências nas políticas e regulamentações globais de privacidade;

- Funcionários do provedor que são mal-intencionados, especialmente os que possuem funções de alto risco como, por exemplo, administradores de sistema;
- Interceptação de dados em trânsito (ataques *man-in-the-middle*).

Sill et al. (2013, p. 46) menciona que os principais objetivos de segurança que o provedor deve ter são:

- Proteger os dados dos clientes contra acesso, divulgação, modificação ou monitoramento não autorizados. Isso inclui o suporte a políticas de gerenciamento de identidade e controle de acesso para usuários autorizados. Capacidade de um cliente disponibilizar o acesso aos seus dados seletivamente para outros usuários;
- Impedir o acesso não autorizado a recursos de infraestrutura. Implementar domínios de segurança com separação lógica entre recursos de computação;
- Implementar APIs padronizadas para interoperabilidade e portabilidade para facilitar a migração dos dados do cliente para outros provedores de nuvem quando necessário;
- Incluir medidas para proteger dispositivos de computação pessoal conectados à Internet por meio da aplicação de *software* de segurança, *firewalls* pessoais e *patch maintenance*;
- Incluir soluções de controle de acesso e detecção e prevenção de intrusão e realizar uma avaliação independente para verificar se as soluções estão instaladas e funcionais;
- Restringir o acesso físico à rede e aos dispositivos;
- Proteger componentes individuais da exploração por meio da implantação de *patch* de segurança; definir como padrão as configurações mais seguras; desabilitando todas as portas e serviços não utilizados;
- Monitoramento de trilhas de auditoria; minimizar os privilégios ao mínimo necessário; usar software antivírus e criptografia de comunicações.
- Definir limites entre o provedor e o cliente para garantir que as responsabilidades de implementar controles de segurança sejam claramente identificadas.
- Implementar APIs padronizadas para interoperabilidade e portabilidade para facilitar a migração dos dados dos consumidores para outros provedores de nuvem quando necessário.

O Sill et al. (2013, p.44) menciona que os sistemas em nuvem normalmente são componentes externos ao setor de TI de uma organização. A necessidade de ter uma integração de segurança sem falhas exige *interfaces* que atuem juntas para autenticação, autorização e proteções de comunicação. Os desafios do gerenciamento de identidade e acesso em diferentes domínios de rede e administração são mais relevantes no sistema em nuvem, pois a implementação desses recursos geralmente não é a mesma utilizada pelo cliente, de onde as informações de identidade se originam. A padronização em áreas como gerenciamento, cópia segura e eficiente em diferentes sistemas ajudam a melhorar os recursos de gerenciamento de identidade em sistemas em nuvem, o que colabora com a segurança. Também é necessário padronizar políticas, processos e controles técnicos que dão suporte aos requisitos de auditoria de segurança, regulamentos e conformidade. É necessário considerar o processo de colaboração entre o cliente e provedores de nuvem, suas funções e o compartilhamento das responsabilidades.

3.1. RESPONSABILIDADES

Quando é feita a implementação de aplicações em servidores próprios o cliente é responsável por toda a estrutura, inclusive a segurança. Na nuvem, como já mencionado, essa responsabilidade é compartilhada entre o cliente e o provedor do serviço, nesse caso é necessário ter claro a divisão das obrigações do cliente e do provedor no quesito segurança. Segundo COGNITIVE CLASS (2021), essa responsabilidade muda conforme o serviço usado na nuvem, por exemplo, PaaS (Plataforma como Serviço), o cliente constrói um aplicativo e o executa na nuvem. Ele é responsável por proteger os aplicativos, a carga de trabalho, os dados, enquanto o provedor é responsável pela gestão da segurança da plataforma, em gerenciar os contêineres, se a aplicação é compatível, o tempo de execução e o isolamento para que cada cliente tenha seu próprio espaço dentro da plataforma. Em IaaS (Infraestrutura como Serviço) o cliente migra cargas de trabalho para a nuvem usando servidores virtuais ou *Bare Metal*, nesse caso ele consegue realizar o controle do sistema operacional, dos servidores virtuais, etc e o provedor é responsável por gerenciar o *hypervisor*, a máquina. No caso de SaaS o provedor administra todos os aplicativos e a segurança dos mesmos, o cliente é responsável pelos dados.

É muito importante entender o modelo que será usado para consumo da nuvem: IaaS, PaaS, ou SaaS porque ele irá definir a responsabilidade de cada um, cliente/provedor, no gerenciamento dos riscos, conformidade das cargas de trabalho e dos dados que são carregados na nuvem. Quanto mais controle o cliente tem, mais responsabilidades ele deve assumir.

Quando se constrói e migra um aplicativo, ou é feita a modernização de um existente, é necessário questionar quais são os dados dessa aplicação? São dados confidenciais? São dados públicos ou dados sensíveis? Quem são os responsáveis pelas informações privadas? Que tipo de aplicativo será construído? Está baseado em contêiner? É uma carga de trabalho que será migrada? Essas informações são importantes para realizar um design seguro. Por exemplo, se é utilizado o serviço de Banco de Dados pode ser usada criptografia, pode-se usar o armazenamento como serviço ou em blocos. Pode ser feito um gerenciamento com uso de chaves para um maior controle, nesse caso o cliente pode usar a própria chave ou uma fornecida pelo provedor, em caso de dados mais sensíveis recomendado usar a própria. É necessário pensar em como proteger os dados em movimento, os que estão em "repouso", somente armazenados e os que estão em uso. Em todas essas possibilidades podem ser usadas chaves.

Na nuvem, COGNITIVE CLASS (2021), pode ser feito o *scanner* das aplicações para buscar vulnerabilidades e definir políticas para garantir que elas estejam seguras. É necessário garantir que quem acessa a aplicação tenha autorização para fazê-lo e ter a certeza de que não se trata de um intruso, esses devem ser impedidos de conseguir acessar através da configuração de *firewall*, controle de acesso a rede ou negação distribuída de proteção de serviço. Mesmo as pessoas que têm autorização para acessar os dados, devem ser revisados quais dados elas podem acessar. É importante gerenciar o acesso aos aplicativos e a carga de trabalho nos dados que são implementados.

A segurança deve ser contínua, deve ser feitos monitoramento para saber se tudo está em conformidade com as políticas criadas. Obter *insights* sobre conformidades e ameaças, criar eventos de segurança, logs de auditoria, logs de fluxo de rede ou sistema. Ademais é necessário ter um plano de ação em caso de vulnerabilidade ou não conformidade, por exemplo, quando a rede é acessada por um endereço IP suspeito, ao realizar o monitoramento e a identificação o cliente poderá realizar o bloqueio do acesso.

Independente do serviço escolhido o provedor é responsável pela segurança física aos *data centers*, controle de acesso, evitar a conexão não autorizada as redes das regiões, entre outros serviços que devem ser definidos em contrato.

3.2. MAPA DOS PADRÕES DE SEGURANÇA

As tabelas 1, 2 e 3, criadas por Sill et al. (2013, p. 52-58), mapeiam os padrões de segurança a serem seguidos por cliente e provedor para várias categorias. Algumas das normas listadas se aplicam a mais de uma categoria e, portanto, são listadas mais de uma vez. Essas tabelas ajudam a atingir padrões de segurança, por exemplo, se for necessário criar um processo para a categoria Autenticação e Autorização pode ser usada as normas mencionadas na coluna Padronização Disponível. O que trará maior segurança para o processo criado.

Tabela 1. Mapa dos Padrões de Segurança

Categoria	Padronização Disponível	Instituto
Autenticação e Autorização	RFC 5246 Secure Sockets Layer (SSL)/ Transport Layer Security (TLS)	IETF
	RFC 3820: X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile	
	RFC5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	
	RFC 5849 OAuth (Open Authorization Protocol)	
	ISO/IEC 9594-8:2008 X.509 Information technology -- Open Systems Interconnection -- The Directory: Public- key and attribute certificate frameworks	ISO/IEC & ITU-T
	ISO/IEC 29115 X.1254 Information technology -- Security techniques -- Entity authentication assurance framework	
	FIPS 181 Automated Password Generator	NIST
	FIPS 190 Guideline for the Use of Advanced Authentication Technology Alternatives	
	FIPS 196 Entity Authentication Using Public Key Cryptography	
		OpenID Authentication
	eXtensible Access Control Markup Language (XACML) Security Assertion Markup Language (SAML)	OASIS

Fonte: adptado de Sill et al., 2013

Tabela 2. Mapa dos Padrões de Segurança

Categoria	Padronização Disponível	Instituto
Confidencialidade	RFC 5246 Secure Sockets Layer (SSL)/ Transport Layer Security (TLS)	IETF
	Key Management Interoperability Protocol (KMIP)	OASIS
	XML Encryption Syntax and Processing	W3C
	FIPS 140-2 Security Requirements for Cryptographic Modules FIPS 185 Escrowed Encryption Standard (EES) FIPS 197 Advanced Encryption Standard (AES)	NIST
	FIPS 188 Standard Security Label for Information Transfer	
Integridade	XML signature (XMLDSig)	W3C
	FIPS 180-4 Secure Hash Standard (SHS) FIPS 186-4 Digital Signature Standard (DSS) FIPS 198-1 The Keyed-Hash Message Authentication Code (HMAC)	NIST
Disponibilidade	ATIS-02000009 Cloud Services Lifecycle Checklist	ATIS
	ISO/PAS 22399:2007 Societal security - Guideline for incident preparedness and operational continuity management	ISO
Gerenciamento de Identidade	X.idmcc Requirement of IdM in Cloud Computing	ITU-T
	FIPS 201-1 Personal Identity Verification (PIV) of Federal Employees and Contractors	NIST
	Service Provisioning Markup Language (SPML) Web Services Federation Language (WS- Federation) Version 1.2 WS-Trust 1.3 Security Assertion Markup Language (SAML)	OASIS
	OpenID Authentication 1.1	OpenID
Monitoramento e Reposta a Incidentes de Segurança	ISO/IEC WD 27035-1 Information technology - Security techniques - Information security incident management - Part 1: Principles of incident management ISO/IEC WD 27035-3 Information technology - Security techniques - Information security incident management - Part 3: Guidelines for CSIRT operations	ISO/IEC
	ISO/IEC WD 27039 Information technology - Security techniques - Selection, deployment and operations of intrusion detection systems	
	ISO/IEC 18180 Information technology - Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2 (NIST IR 7275)	
	X.1500 Cybersecurity information exchange techniques X.1520: Common vulnerabilities and exposures X.1521 Common Vulnerability Scoring System	ITU-T
	PCI Data Security Standard	PCI

Fonte: adaptado de Sill et al., 2013

Tabela 3. Mapa dos Padrões de Segurança

Categoria	Padronização Disponível	Instituto
	Cloud Controls Matrix Version 1.3	CSA
Controles de Segurança	ISO/IEC 27001:2005 Information Technology – Security Techniques Information Security Management Systems Requirements	ISO/IEC
	ISO/IEC WD TS 27017 Information technology -- Security techniques -- Information security management - Guidelines on information security controls for the use of cloud computing services based on ISO/IEC 27002	
	ISO/IEC 27018 Code of Practice for Data Protection Controls for Public Cloud Computing Services	
	ISO/IEC 1 WD 27036-4 Information technology – Security techniques – Information security for supplier relationships – Part 4: Guidelines for security of cloud services	
Gerenciamento de Políticas de Segurança	ATIS-02000008 Trusted Information Exchange (TIE)	ATIS
	FIPS 199 Standards for Security Categorization of Federal Information and Information Systems	NIST
	FIPS 200 Minimum Security Requirements for Federal Information and Information Systems	
	ISO/IEC 27002 Code of practice for information security management	ISO/IEC
	eXtensible Access Control Markup Language (XACML)	OASIS

Fonte: adaptado de Sill et al., 2013

3.3. SEIS ETAPAS PARA UMA COMPUTAÇÃO EM NUVEM MAIS SEGURA

Jillson e Hasty (2020) fornecem seis dicas para tornar o uso da nuvem mais segura tanto para o provedor como para o cliente.

3.3.1. Aproveitar os Recursos de Segurança oferecidos pelas empresas de serviço em nuvem

Os provedores de nuvem oferecem orientação detalhada sobre seus controles de segurança e como configurar seus serviços de maneira mais segura, mas cabe ao cliente entender as opções e definir essas configurações da maneira mais adequada ao seu negócio. A configuração da segurança na nuvem exige que se tome decisões ponderadas, alinhadas com a confidencialidade dos dados que é armazenado e como será usado. Além disso, é necessário avaliar cuidadosamente sobre quem, na empresa, precisa ter acesso a quais dados. A menos que os funcionários tenham um motivo comercial legítimo, eles não devem ter acesso aos recursos de nuvem. Deve-se exigir autenticação multifator e senhas fortes para proteger contra o risco de acesso

não autorizado. Nunca codificar senhas em aplicativos baseados em nuvem ou código-fonte.

3.3.2. Fazer inventários regulares do que mantém na nuvem

Quer armazene dados na nuvem, ou em sua rede ou em um arquivo, o cliente não pode manter os dados seguros se não souber onde eles estão. É por isso que o inventário atualizado é essencial para o gerenciamento de dados. Muitos serviços em nuvem fornecem ferramentas, por exemplo, painéis ou consoles de gerenciamento, exatamente para essa finalidade. Mas não basta configurá-lo e esquecê-lo. Além de ficar por dentro de onde estão todo e qualquer dado, o cliente deve certificar-se de que suas configurações de segurança e direitos de acesso permaneçam consistentes com a confidencialidade do que foi armazenada. À medida que são adicionados novos dados que podem exigir mais proteção, é necessário reavaliar as configurações de segurança. Realizar testes por configurações incorretas ou outras falhas de segurança que possam comprometer os dados e manter arquivos de log para monitorar continuamente os repositórios na nuvem.

3.3.3. Não armazenar informações pessoais quando não for necessário

Uma vantagem do armazenamento em nuvem é que geralmente é mais barato do que outros métodos. Mas a lista de coisas consideradas essenciais tende a se expandir na proporção direta de quanto espaço de armazenamento está disponível. Ao realizar o inventário do que se mantém na nuvem, deve-se resistir à tentação de manter os dados *just because*. Em vez disso, o cliente deve perguntar-se: “Temos uma necessidade legítima de armazenar essas informações?” Se a resposta for não, descarte-o com segurança. Ninguém pode violar o que não se tem.

3.3.4. Considerar encriptografar dados raramente usados

Existem alguns clientes que dizem “Há algumas informações que não preciso acessar regularmente – *backups*, por exemplo, mas preciso retê-las.” Como parte de uma abordagem de defesa profunda à segurança, considerar criptografar os dados em repouso, ou seja, que não estão em uso. Se esses dados contiverem informações confidenciais, fazer a criptografia. Isso é um princípio básico de segurança, independentemente de onde estejam armazenados.

3.3.5. Prestar atenção aos avisos confiáveis

Alguns provedores de nuvem oferecem ferramentas automatizadas para lembrar o cliente sobre repositórios de nuvem abertos à Internet. Outros podem entrar em contato com os clientes com avisos como esse. Em outros casos, pesquisadores de segurança podem entrar em contato com empresas quando encontram dados expostos *online*. Se um cliente receber um desses avisos é necessário prestar atenção. Investigar os repositórios na nuvem e verificar as configurações de segurança.

3.3.6. A segurança é responsabilidade do cliente

Usar serviços em nuvem não significa que a segurança pode ser terceirizada. Durante todo o ciclo de vida dos dados na posse do cliente, a segurança continua sendo responsabilidade dele. Mesmo que o cliente confie nas ferramentas de segurança do provedor de nuvem, ele ainda deve ter um programa escrito de segurança de dados que estabeleça o processo da empresa para proteger as informações pessoais dos consumidores e pessoas em sua equipe com conhecimento sobre manutenção, monitoramento, teste e atualização desse programa. O cliente precisa revisar seus contratos de nuvem cuidadosamente para definir suas expectativas e estabelecer claramente quem é o principal responsável pelo quê. Em última análise o principal responsável pelos dados é o próprio cliente.

4. NUVEM NA PRÁTICA

Nesse capítulo serão apresentados alguns conceitos da nuvem através da execução do Kubernetes.

A ferramenta principal que foi utilizada foi o Minikube que “facilita a execução do Kubernetes localmente” (ESTEVEZ 2018). Os comandos para execução são padronizados por esse motivo as práticas feitas nesse trabalho podem ser encontradas em documentações e repositórios na internet. O objetivo principal foi mostrar os conceitos sobre nuvem usando ferramentas que possuem um padrão de comandos a serem executados e demonstrar os resultados obtidos após a criação e teste de cada etapa. Todas as etapas demonstradas foram criadas e testadas localmente, possibilitando o aprendizado e melhor entendimento de cada conceito.

A prática foi realizada em um computador MacOs com sistema operacional Monterey. Foi necessário instalar e configurar o Docker Desktop on Mac (DOCKER INC, 2021) e o programa minikube-darwin-amd64 com as instruções abaixo fornecidas pelo instrutor da GEEK UNIVERSITY (2022). Para algumas práticas foi utilizada a linguagem YAML. “O YAML é uma linguagem de serialização de dados muito usada na escrita de arquivos de configuração” (RED HAT, INC, 2021)

1. *Download* do *kubectl*, arquivo adicionado ao *path* do sistema, liberando a permissão e arquivo movido de pasta:

```
curl -LO https://storage.googleapis.com/Kubernetes-release/release/$(curl -s https://storage.googleapis.com/Kubernetes-release/release/stable.txt)/bin/darwin/amd64/kubectl
```

```
chmod +x ./kubectl
```

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

2. Verificou-se a instalação:

```
kubectl version --client
```

3. Executado o instalador do *Minikube*:

```
sudo install minikube-darwin-amd64 /usr/local/bin/minikube
```

4. Configurado o Docker como *driver* padrão para o Kubernetes

```
minikube config set driver docker
```

5. Inicializado o *cluster*

```
minikube start
```

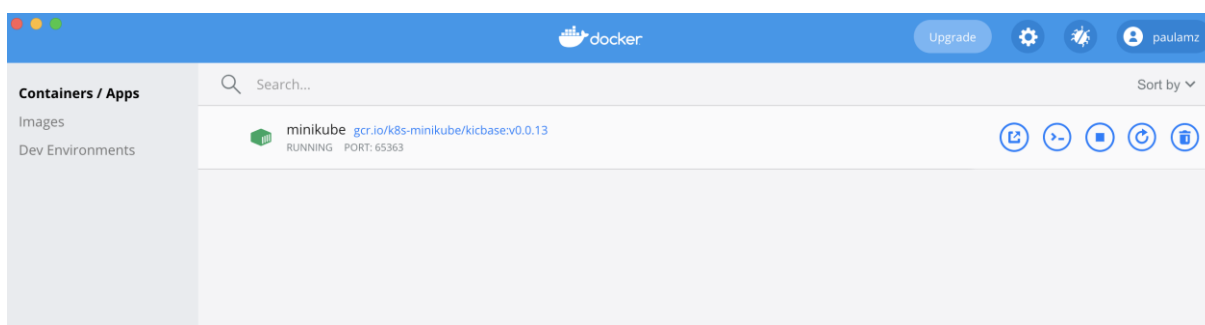
Após o último comando, foi necessário aguardar a mensagem *Done* para começar a usar o *cluster*.

Tela do terminal com seu respectivo resultado:

```
pmezzomo@Paulas-MBP ~ % minikube start
🤗 minikube v1.14.2 on Darwin 12.3.1
🌟 Using the docker driver based on existing profile
👍 Starting control plane node minikube in cluster minikube
🏃 Updating the running docker "minikube" container ...
🐳 Preparing Kubernetes v1.19.2 on Docker 19.03.8 ...
🔍 Verifying Kubernetes components...
🌟 Enabled addons: storage-provisioner, default-storageclass

! /usr/local/bin/kubectl is version 1.23.3, which may have incompatibilites wi
th Kubernetes 1.19.2.
💡 Want kubectl v1.19.2? Try 'minikube kubectl -- get pods -A'
🏆 Done! kubectl is now configured to use "minikube" by default ←
```

A criação do *cluster* também pode ser visualizada no aplicativo Docker Desktop



Após a instalação foram testados os comandos e instalação de algumas aplicações prontas cuja imagens estão disponíveis *online*, seguindo o instrutor da GEEK UNIVERSITY (2022). No capítulo sobre Contêineres foi explicado o que é uma imagem.

Os primeiros testes foram realizados com a instalação de um servidor web Nginx no cluster Kubernetes.

Com o comando `kubectl run nginx --image nginx` foi criado um *pod* do servidor Nginx e com o comando `kubectl get pods` é possível verificar os pods criados. Como pode ser visto o servidor foi criado em seguida ao comando. A aplicação já está disponível há 6 segundos - coluna *Age*, demonstrando a rapidez do *deploy* na nuvem. Também pode ser visto a quantidade de *n pods* para *n contêineres*.

Tela do terminal com seu respectivo resultado:

```
pmezozomo@Paulas-MBP ~ % kubectl run nginx --image nginx
pod/nginx created
pmezozomo@Paulas-MBP ~ % kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
nginx         0/1    ContainerCreating   0           6s
```

Com o comando `kubectl get pods -o wide` ou `kubectl describe pod nginx` é possível ver mais detalhes, como por exemplo, nome do `pod`, endereço `node`, IP do `node`, data de início, os eventos, entre outros dados:

Tela do terminal com seu respectivo resultado:

```
pmezozomo@Paulas-MBP ~ % kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP           NODE          NOMINATED NODE   READINESS GATES
nginx         1/1    Running   0           106s  172.17.0.3  minikube     <none>           <none>
pmezozomo@Paulas-MBP ~ % kubectl describe pod nginx
Name:          nginx
Namespace:    default
Priority:      0
Node:         minikube/192.168.49.2
Start Time:   Wed, 20 Apr 2022 23:02:14 -0300
Labels:       run=nginx
Annotations:  <none>
Status:       Running
IP:          172.17.0.3
IPs:
IP: 172.17.0.3
Containers:
  nginx:
    Container ID:  docker://5ecfa72526f529f1cddcc936199bb4d5666d0751568060331607961ce75c3098
    Image:         nginx
    Image ID:     docker-pullable://nginx@sha256:6d701d83f2a1bb99efb7e6a60a1e4ba6c495bc5106c91709b0560d13a9bf8fb6
    Port:         <none>
    Host Port:    <none>
    State:        Running
      Started:    Wed, 20 Apr 2022 23:02:31 -0300
    Ready:        True
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-4phz1 (ro)
Conditions:
  Type           Status
  Initialized     True
  Ready           True
  ContainersReady True
  PodScheduled    True
Volumes:
  default-token-4phz1:
    Type: Secret (a volume populated by a Secret)
    SecretName: default-token-4phz1
    Optional: false
QoS Class:   BestEffort
Node-Selectors: <none>
Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s
              node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type     Reason      Age   From          Message
  ----     -
  Normal   Scheduled   112s  default-scheduler  Successfully assigned default/nginx to minikube
  Normal   Pulling     111s  kubelet         Pulling image "nginx"
  Normal   Pulled      95s   kubelet         Successfully pulled image "nginx" in 15.655662s
  Normal   Created     95s   kubelet         Created container nginx
  Normal   Started     95s   kubelet         Started container nginx
```

No próximo teste demonstrou-se o conceito de disponibilidade para isso ela deve ter no mínimo duas réplicas, pois se uma falhar a outra continuará funcionando.

Em um provedor de nuvem, cada réplica pode ser hospedada em uma região distinta e pode ser feito o *Load Balancer*, conceito explicado no capítulo sobre Rede.

Para essa demonstração foi utilizado o *ReplicaSets* e YAML. Existe um padrão para ser utilizado com a linguagem YAML, por esse motivo, foi utilizado o código disponibilizado pelo instrutor da GEEK UNIVERSITY (2022):

Arquivo *replication.yml*:

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: tcc-cloud
  labels:
    app: frontend
spec:
  template:
    metadata:
      name: nginx-pod
      labels:
        env: production
    spec:
      containers:
        - name: nginx-container
          image: nginx
  selector:
    matchLabels:
      env: production
  replicas: 2
```

Comando `kubectl create -f replication.yml`. Foram criadas duas réplicas como pode ser visto na coluna *Name*.

Tela do terminal com seu respectivo resultado:

```
[pmezzomo@Paulas-MBP replicasets % kubectl create -f replication.yml
replicaset.apps/tcc-cloud created
[pmezzomo@Paulas-MBP replicasets % kubectl get pods
NAME                READY   STATUS              RESTARTS   AGE
tcc-cloud-mrx2r     0/1     ContainerCreating   0           4s
tcc-cloud-wm7pb     0/1     ContainerCreating   0           4s
```

Em seguida, foi apagado o Pod `tcc-cloud-mrx2r` e automaticamente foi criada a nova réplica com nome `tcc-cloud-nxwfb`.

Tela do terminal com seu respectivo resultado:

```
pmezzomo@Paulas-MBP replicasets % kubectl get pods
NAME                READY   STATUS              RESTARTS   AGE
tcc-cloud-mrx2r     0/1     ContainerCreating   0           4s ←
tcc-cloud-wm7pb     0/1     ContainerCreating   0           4s
pmezzomo@Paulas-MBP replicasets % kubectl delete pod tcc-cloud-mrx2r
pod "tcc-cloud-mrx2r" deleted
pmezzomo@Paulas-MBP replicasets % kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
tcc-cloud-nxwfb     1/1     Running   0           15s ←
tcc-cloud-wm7pb     1/1     Running   0          2m10s
```

Com o comando `kubectl describe replicaset tcc-cloud` é possível verificar informações do *ReplicaSet*, por exemplo, no *Events* é possível ver todos os *pods* criados mesmo que ele tenha sido apagado, no exemplo, foi removido o `tcc-cloud-mrx2r`:

Tela do terminal com seu respectivo resultado:

```
pmezzomo@Paulas-MBP replicasets % kubectl describe replicaset tcc-cloud
Name:                tcc-cloud
Namespace:           default
Selector:            env=production
Labels:              app=frontend
Annotations:         <none>
Replicas:            2 current / 2 desired
Pods Status:        2 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:            env=production
  Containers:
    nginx-container:
      Image:          nginx
      Port:           <none>
      Host Port:     <none>
      Environment:   <none>
      Mounts:        <none>
      Volumes:       <none>
Events:
  Type     Reason             Age   From                    Message
  ----     -
  Normal   SuccessfulCreate   5m30s replicaset-controller  Created pod: tcc-cloud-mrx2r
  Normal   SuccessfulCreate   5m30s replicaset-controller  Created pod: tcc-cloud-wm7pb
  Normal   SuccessfulCreate   3m35s _replicaset-controller  Created pod: tcc-cloud-nxwfb
```

Também é possível fazer um *scale*, aumentar (*scale up*) ou diminuir (*scale down*) o número de réplicas conforme o uso. Se existe uma alta demanda aumenta a quantidade, do contrário reduz.

No exemplo foram utilizados os comandos direto no terminal, foram solicitadas quatro réplicas e depois reduzidas a uma réplica.

Tela do terminal com seu respectivo resultado:

```
pmezozomo@Paulas-MBP replicasets % kubectl scale replicaset tcc-cloud --replicas=4 ←
replicaset.apps/tcc-cloud scaled
pmezozomo@Paulas-MBP replicasets % kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
tcc-cloud-9hgbt 0/1     ContainerCreating  0           4s
tcc-cloud-hcbv5 0/1     ContainerCreating  0           4s
tcc-cloud-nxwfb 1/1     Running            0          10m
tcc-cloud-wm7pb 1/1     Running            0          12m
pmezozomo@Paulas-MBP replicasets % kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
tcc-cloud-9hgbt 1/1     Running            0          16s
tcc-cloud-hcbv5 1/1     Running            0          16s
tcc-cloud-nxwfb 1/1     Running            0          10m
tcc-cloud-wm7pb 1/1     Running            0          12m
pmezozomo@Paulas-MBP replicasets % kubectl scale replicaset tcc-cloud --replicas=1 ←
replicaset.apps/tcc-cloud scaled
pmezozomo@Paulas-MBP replicasets % kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
tcc-cloud-9hgbt 0/1     Terminating      0          3m58s
tcc-cloud-nxwfb 0/1     Terminating      0          14m
tcc-cloud-wm7pb 1/1     Running            0          16m
pmezozomo@Paulas-MBP replicasets % kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
tcc-cloud-wm7pb 1/1     Running            0          16m
```

Foi feito um novo *deploy* para a aplicação Nginx utilizando *NodePort*. Primeiro foi criada a aplicação e em seguida criado o *service* para conexão com a internet.

Arquivo frontend-dp.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-dp
  labels:
    app: frontend-app
    type: frontend
spec:
  template:
    metadata:
      name: frontend-pod
      labels:
        app: frontend-app
        type: frontend
    spec:
      containers:
        - name: nginx-container
          image: nginx
  selector:
    matchLabels:
      type: frontend
  replicas: 6
```

Tela do terminal com seu respectivo resultado:

```
pmezzomo@Paulas-MBP nginx % kubectl create -f frontend-dp.yaml --save-config
deployment.apps/frontend-dp created
```

Arquivo frontend-svc.yaml (conexão na porta 80):

```
apiVersion: v1
kind: Service
metadata:
  name: frontend-svc
spec:
  selector:
    type: frontend
  ports:
    - name: http
      port: 80
      nodeport: 30080
  type: NodePort
```

Tela do terminal com seu respectivo resultado:

```
pmezzomo@Paulas-MBP nginx % kubectl create -f frontend-svc.yaml --save-config --validate=false
service/frontend-svc created
```

```
pmezzomo@Paulas-MBP nginx % kubectl get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/frontend-dp-79fdffd664-7fxmj	1/1	Running	0	64s
pod/frontend-dp-79fdffd664-8mtnd	1/1	Running	0	64s
pod/frontend-dp-79fdffd664-f9bbp	1/1	Running	0	64s
pod/frontend-dp-79fdffd664-hgdqd	1/1	Running	0	64s
pod/frontend-dp-79fdffd664-x8hpr	1/1	Running	0	64s
pod/frontend-dp-79fdffd664-xnbt6	1/1	Running	0	64s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/frontend-svc	NodePort	10.105.237.231	<none>	80:31232/TCP	40s
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	105s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/frontend-dp	6/6	6	6	64s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/frontend-dp-79fdffd664	6	6	6	64s

Logo após, foi necessário verificar qual o IP utilizado para acesso da aplicação via navegador. Nota-se que não foi utilizado o IP do Cluster, mas um pertencente a mesma classe da rede local.

Com *ifconfig* é possível visualizar o IP local que pertence a classe A.

Tela do terminal com seu respectivo resultado:

```
pmezzomo@Paulas-MBP nginx % ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    options=1203<RXCSUM, TXCSUM, TXSTATUS, SW_TIMESTAMP>
    inet 127.0.0.1 netmask 0xff000000
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
    nd6 options=201<PERFORMNUD,DAD>
```

Tela do terminal com seu respectivo resultado:

```
pmezzomo@Paulas-MBP nginx % minikube ip
127.0.0.1
pmezzomo@Paulas-MBP nginx % minikube service frontend-svc --url
🐳 Starting tunnel for service frontend-svc.
-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | frontend-svc | | http://127.0.0.1:61481 |
|-----|-----|-----|-----|
http://127.0.0.1:61481
! Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

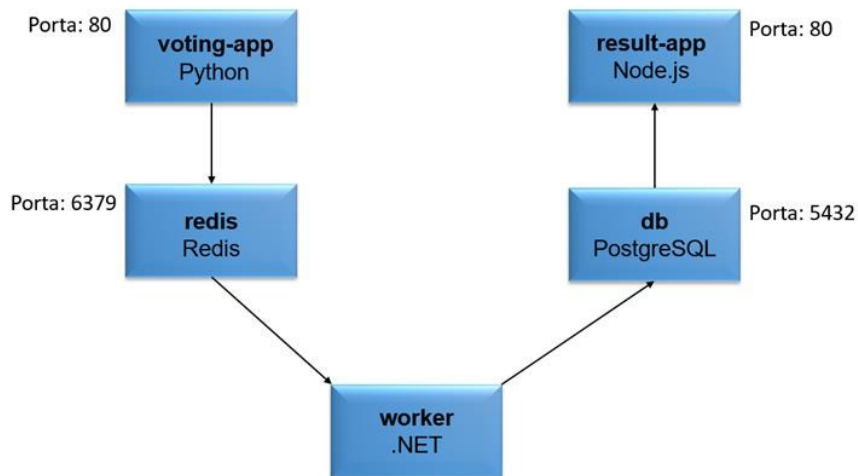
Tela do navegador com o respectivo resultado:



Nessa próxima etapa, foi simulado o uso de microsserviços para uma aplicação *web*, com a simulação de uma aplicação de voto *online*. Foi utilizado a imagem de aplicação pronta, pois o objetivo principal, como já mencionado, é apresentar conceitos de nuvem através da execução do Kubernetes localmente. Esse exemplo pode ser encontrado no Github (2022) do Docker. As imagens da aplicação estão armazenadas e foram baixadas do *kodekloud/examplevotingapp_worker:v1* conforme pode ser visto no código YAML.

Para exibição no navegador do voto *online*, conforme figura 20, foi utilizada uma imagem com Python para exibir as aplicações no *front-end* porta 80 e o usuário votar na opção cão ou gato. No momento dessa votação, a informação é salva no banco de dados “em memória”, usado o Redis na porta 6379. Usada a aplicação .NET que consulta a aplicação Redis para ver se houve votação. Se houver a votação, a aplicação salva a informação para o banco de dados PostgreSQL na porta 5432 que por sua vez envia o dado para a aplicação Node.js que exibe o resultado da votação para o usuário na porta 80. Foram criados módulos conectados.

Figura 20. Aplicação em Microsserviços



Fonte: adaptado de GEEK UNIVERSITY (2022)

Primeiro foi criado o *namespace* “vote” para organizar as aplicações em um mesmo *cluster*. Código executado via terminal:

```

apiVersion: v1
kind: Namespace
metadata:
  name: vote
  
```

Tela do terminal com seu respectivo resultado:

```

[pmezzomo@Paulas-MBP cloud % kubectl create -f namespaces --save-config
namespace/vote created
[pmezzomo@Paulas-MBP cloud % kubectl get namespaces
NAME                STATUS    AGE
default              Active    3m15s
kube-node-lease     Active    3m16s
kube-public          Active    3m16s
kube-system          Active    3m17s
vote                 Active    101s
  
```

Em seguida, realizado os *deploys* de cada aplicação:

Python (vote):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: vote
    name: vote
  namespace: vote
spec:
  replicas: 2
  selector:
    matchLabels:
      app: vote
  template:
    metadata:
      labels:
        app: vote
    spec:
      containers:
        - image: kodekloud/examplevotingapp_vote:v1
          name: vote
          ports:
            - containerPort: 80
              name: vote
```

Redis:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: redis
    name: redis
  namespace: vote
spec:
  replicas: 2
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
        - image: redis:alpine
          name: redis
          ports:
            - containerPort: 6379
              name: redis
```

```
volumeMounts:  
- mountPath: /data  
  name: redis-data  
volumes:  
- name: redis-data  
  emptyDir: {}
```

.Net (worker):

```
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  labels:  
    app: worker  
    name: worker  
  namespace: vote  
spec:  
  replicas: 2  
  selector:  
    matchLabels:  
      app: worker  
  template:  
    metadata:  
      labels:  
        app: worker  
    spec:  
      containers:  
        - image: kodekloud/examplevotingapp_worker:v1  
          name: worker
```

PostgreSQL:

```
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  labels:  
    app: db  
    name: db  
  namespace: vote  
spec:  
  replicas: 2  
  selector:  
    matchLabels:  
      app: db  
  template:  
    metadata:  
      labels:  
        app: db  
    spec:  
      containers:  
        - image: postgres:9.4
```

```

name: postgres
env:
- name: POSTGRES_USER
  value: postgres
- name: POSTGRES_PASSWORD
  value: postgres
ports:
- containerPort: 5432
  name: postgres
volumeMounts:
- mountPath: /var/lib/postgresql/data
  name: db-data
volumes:
- name: db-data
  emptyDir: {}

```

Node.js (result):

```

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: result
    name: result
  namespace: vote
spec:
  replicas: 2
  selector:
    matchLabels:
      app: result
  template:
    metadata:
      labels:
        app: result
    spec:
      containers:
      - image: kodekloud/examplevotingapp_result:v1
        name: result
        ports:
        - containerPort: 80
          name: result

```

Tela do terminal com seu respectivo resultado:

```

[pmezzomo@Paulas-MBP cloud % kubectl create -f deployments --save-config
deployment.apps/db created
deployment.apps/redis created
deployment.apps/result created
deployment.apps/vote created
deployment.apps/worker created

```

Para comunicação entre as aplicações foi necessário criar o *service*. Somente não foi criado para a aplicação .NET porque os outros serviços não precisam se comunicar com ela, assim que ela não precisa ficar disponível.

Python (vote):

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: vote
    name: vote
    namespace: vote
spec:
  type: NodePort
  ports:
  - name: "vote-service"
    port: 5000
    targetPort: 80
    nodePort: 31000
  selector:
    app: vote
```

Redis:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: redis
    name: redis
    namespace: vote
spec:
  type: ClusterIP
  ports:
  - name: "redis-service"
    port: 6379
    targetPort: 6379
  selector:
    app: redis
```

PostgreSQL:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: db
    name: db
    namespace: vote
spec:
```

```

type: ClusterIP
ports:
- name: "db-service"
  port: 5432
  targetPort: 5432
selector:
  app: db

```

Node.js (result):

```

apiVersion: v1
kind: Service
metadata:
  labels:
    app: result
    name: result
  namespace: vote
spec:
  type: NodePort
  ports:
  - name: "result-service"
    port: 5001
    targetPort: 80
    nodePort: 31001
  selector:
    app: result

```

Tela do terminal com seu respectivo resultado:

```

pmezzomo@Paulas-MBP cloud % kubectl create -f services/ --save-config
service/db created
service/redis created
service/result created
service/vote created

```

Através do *namespace* criado “vote” é possível ver o *status* da criação de cada aplicação e a quantidade de réplicas. Também é possível visualizar o cluster-IP e as portas.

Tela do terminal com seu respectivo resultado:

```
pmezozomo@Paulas-MBP cloud % kubectl get all -n vote
NAME                                READY   STATUS    RESTARTS   AGE
pod/db-684b9b49fd-rhbz7             1/1     Running   0           2m29s
pod/db-684b9b49fd-rwqfm             1/1     Running   0           2m29s
pod/redis-67db9bd79b-5clgv          1/1     Running   0           2m29s
pod/redis-67db9bd79b-gsfgn          1/1     Running   0           2m29s
pod/result-77f68799c4-fscrn         1/1     Running   0           2m29s
pod/result-77f68799c4-rn4xf         1/1     Running   0           2m29s
pod/vote-79787c6c8b-gdfvj           1/1     Running   0           2m29s
pod/vote-79787c6c8b-khj2v          1/1     Running   0           2m29s
pod/worker-78b9ff59fc-f6gfv         1/1     Running   0           2m29s
pod/worker-78b9ff59fc-rbrhj         1/1     Running   0           2m29s
```

```
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
service/db          ClusterIP      10.107.123.80   <none>           5432/TCP         2m23s
service/redis       ClusterIP      10.104.214.227 <none>           6379/TCP         2m23s
service/result      NodePort       10.111.227.182 <none>           5001:31001/TCP   2m23s
service/vote        NodePort       10.103.30.57   <none>           5000:31000/TCP   2m23s
```

```
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/db  2/2     2             2           2m30s
deployment.apps/redis  2/2     2             2           2m30s
deployment.apps/result  2/2     2             2           2m30s
deployment.apps/vote  2/2     2             2           2m30s
deployment.apps/worker  2/2     2             2           2m30s
```

```
NAME                DESIRED   CURRENT   READY   AGE
replicaset.apps/db-684b9b49fd  2         2         2       2m30s
replicaset.apps/redis-67db9bd79b  2         2         2       2m30s
replicaset.apps/result-77f68799c4  2         2         2       2m30s
replicaset.apps/vote-79787c6c8b  2         2         2       2m30s
replicaset.apps/worker-78b9ff59fc  2         2         2       2m30s
```

Ao finalizar as configurações é possível acessar o *site* para realizar a votação e outro para ver o resultado.

Executados os comandos abaixo no terminal para ter acesso a URL

Tela do terminal com seus respectivos resultados:

```
pmezozomo@Paulas-MBP cloud % minikube service vote --url -n vote
🔗 Starting tunnel for service vote.
```

NAMESPACE	NAME	TARGET PORT	URL
vote	vote		http://127.0.0.1:59762

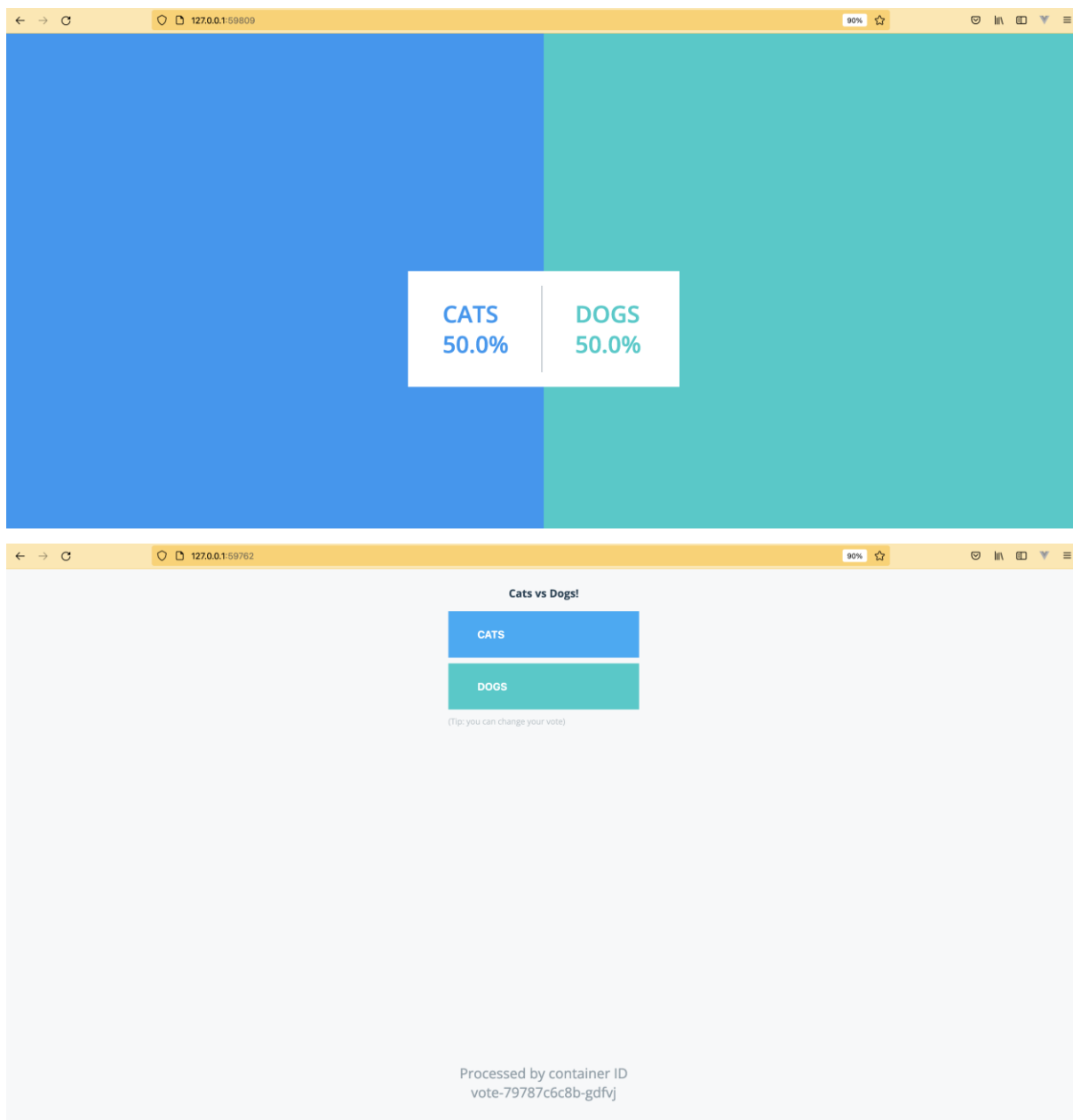
```
http://127.0.0.1:59762
! Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

```
pmezozomo@Paulas-MBP cloud % minikube service result --url -n vote
🔗 Starting tunnel for service result.
```

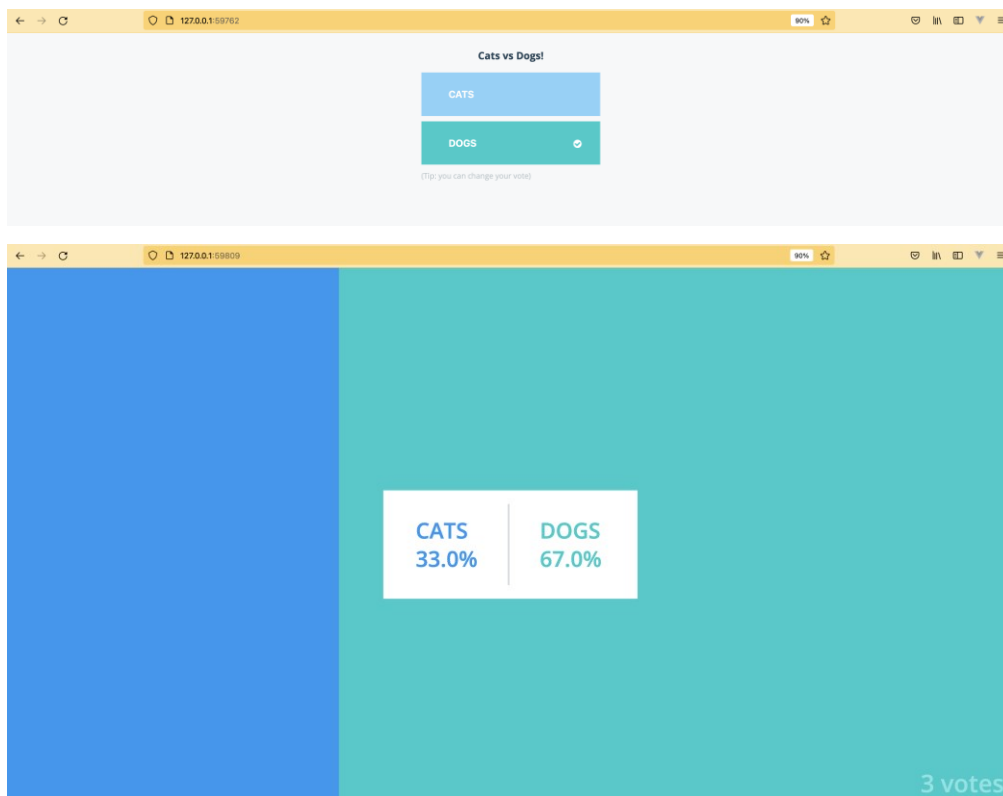
NAMESPACE	NAME	TARGET PORT	URL
vote	result		http://127.0.0.1:59809

```
http://127.0.0.1:59809
! Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```


Tela do navegador para votação e a de resultado antes da primeira votação:



Tela do navegador com a votação realizada e a outra com o resultado:



A prática realizada pode ser exemplificada na figura 21, onde é demonstrado que um *cluster* do Kubernetes é criado, depois implementado um aplicativo nesse *cluster*, são feitas verificações das informações, para alguns casos a aplicação é exposta para uma rede externa. E, no final, é possível realizar a escalabilidade e atualização do aplicativo.

Figura 21. Módulos Básico do Kubernetes



Fonte: Kubernetes (2022)

CONCLUSÃO

Com o aumento dos dados e informações processadas tornou-se necessário utilizar computadores com alta capacidade de processamento para acesso à essas informações em tempo real e ao mesmo tempo reduzir os custos desse uso, o que fez com que muitas empresas migrassem para nuvem. Mas para migrar é necessário revisar a característica de cada empresa, que são únicas. Nesse trabalho, foi possível ver os prós, contras e o que considerar antes de realizar uma migração para que ela seja bem-sucedida. Pontos que também podem ser considerados na construção de uma aplicação direto na nuvem. Também foram vistas as características essenciais de um nativo de nuvem.

Como pôde ser visto Nuvem é um conceito antigo, mas somente nos últimos três anos ele tornou-se mais conhecido. Seu uso tem crescido de maneira exponencial e há carência de profissionais especializados. A falta de conhecimentos prévios ou por muitas vezes não saber como começar a estudar ou até mesmo excesso de informações, podem dificultar a busca do aprendizado sobre esse tema. Com intuito de compartilhar conhecimento e tornar o aprendizado mais acessível esse trabalho foi escrito reunindo os fundamentos da nuvem de uma maneira organizada e objetiva. Com ele foi possível aprender as características essenciais da nuvem, os tipos, a infraestrutura, os benefícios, como funciona a segurança e normas para padronização que podem ser usadas por clientes e provedores, conceito de máquinas virtuais e contêineres.

Ao terminar, o leitor será capaz de ter entendido o funcionamento da nuvem, os principais serviços que podem ser encontrados em qualquer provedor, ter visto na prática o funcionamento de alguns desses conceitos através do uso do Kubernetes e Contêineres. A parte prática consolidou o que foi apresentado ao longo do trabalho. Também, conseguirá entender algumas das idéias mais complexas, tais como: definição de Microsserviços e seus benefícios, Kubernetes e seu funcionamento, os diversos tipos de armazenamento que podem ser encontrados na Nuvem, entre outros tópicos de considerável importância.

Conclui-se que a partir desse momento é possível avançar nos estudos sobre a Nuvem e aprofundar cada um dos temas aqui apresentados.

REFERÊNCIAS

4D DATA CENTERES LTD. **What are IaaS, PaaS, and SaaS, and how do they impact Cloud?** 2020. Disponível em: <https://www.4d-dc.com/insight/what-are-iaas-paas-and-saas-and-how-do-they-impact-cloud>. Acesso em: 31 nov. 2021.

ABRAHOSTING - ASSOCIADO. **IOPS**: saiba o que é e por que são importantes. 2022. Disponível em: <https://blog.eveo.com.br/iops>. Acesso em: 26 mar. 2022.

ARUNDEL, John; DOMINGUS, Justin. **DevOps nativo de nuvem com Kubernetes**: como construir, implantar e escalar aplicações modernas na nuvem. Edição 1. São Paulo: Novatec, 2019. 384 p.

AUGUSTO, Henrique. **SaaS, PaaS e IaaS**: os serviços de computação em nuvem. os serviços de computação em nuvem. 2012. Disponível em: <https://blog.qinetwork.com.br/saas-paas-iaas-os-servicos-de-computacao-em-nuvem/>. Acesso em: 30 abr. 2022.

BADGER, L. et. al. **Cloud Computing Synopsis and Recommendations**: Recommendations of the National Institute of Standards and Technology. Special Publication 800-146. Estados Unidos: NIST, 2012. p. 9-4.

COGNITIVE CLASS. **Introduction to cloud**. 2021. Disponível em: <https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0101EN+2020T1/course/>. Acesso em: 26 abr. 2022.

CONVERGÊNCIA DIGITAL. **Cloud computing**: mercado global de nuvem vai crescer 55% até 2022. Mercado global de nuvem vai crescer 55% até 2022. 2019. Disponível em: <https://www.convergenciadigital.com.br/Cloud-Computing/Mercado-global-de-nuvem-vai-crescer-55%25-ate-2022-52237.html?UserActiveTemplate=mobile%2Csite%3E..> Acesso em: 29 set. 2021.

CORBETT, Xavier. **Introduction to cloud computing**. 2022. Disponível em: <https://ibm-learning.udemy.com/course/introduction-to-cloud-computing/learn/lecture/4430914?start=0#overview>. Acesso em: 19 abr. 2022.

DELL INC. **A História e o Futuro da Computação em Nuvem**. 2022. Disponível em: https://www.dell.com/learn/br/pt/brbsdt1/sb360/social_cloud. Acesso em: 26 jun. 2022.

DOCKER INC. **Install Docker Desktop on Mac**. 2021. Disponível em: <https://docs.docker.com/desktop/mac/install/>. Acesso em: 26 abr. 2022.

EMORY UNIVERSITY'S GOIZUETA BUSSINESS SCHOOL. **Ramnath K Chellappa**: professor of information systems & operations management; associate dean & academic director, ms in business analytics. Professor of Information Systems & Operations Management; Associate Dean & Academic Director, MS in Business Analytics. 2021. Disponível em:

<https://goizueta.emory.edu/faculty/profiles/ramnath-k-chellappa>. Acesso em: 07 nov. 2021.

ESTEVEES, Guilherme. **Minikube: ame-o ou deixe-o**. Ame-o ou deixe-o. 2018. Disponível em: <https://blog.getupcloud.com/minikube-ame-o-ou-deixe-o-dc18fc7cb993>. Acesso em: 05 jun. 2022.

GARTNER, INC. AND/OR ITS AFFILIATES. **Gartner forecasts worldwide public cloud end-user spending to grow 18% in 2021**. 2020. Disponível em: <https://www.gartner.com/en/newsroom/press-releases/2020-11-17-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-18-percent-in-2021>. Acesso em: 03 nov. 2021.

GEEK UNIVERSITY. **Orquestração de containers com Kubernetes**. 2022. Disponível em: <https://ibm-learning.udemy.com/course/orquestracao-de-containers-com-Kubernetes/learn/lecture/22504282#overview>. Acesso em: 25 abr. 2022.

GITHUB, Inc. **GuniversityBR: example-voting-app**. example-voting-app. 2022. Disponível em: <https://github.com/guniversityBR/example-voting-app>. Acesso em: 19 maio 2022.

IBM. **IBM cloud file storage**. Disponível em: <https://www.ibm.com/br-pt/cloud/file-storage>. Acesso em: 26 mar. 2022.

IBM. **Segurança de dados e princípios de privacidade da IBM**. 2020. Disponível em: <https://www.ibm.com/support/customer/csol/terms?id=Z126-7745&lc=pt-br#detail-document>. Acesso em: 21 abr. 2022.

IBM CLOUD EDUCATION. **Armazenamento em bloco**. 2019. Disponível em: <https://www.ibm.com/br-pt/cloud/learn/block-storage>. Acesso em: 26 mar. 2022.

IBM CLOUD EDUCATION. **Armazenamento em cloud**. 2019. Disponível em: <https://www.ibm.com/br-pt/cloud/learn/cloud-storage>. Acesso em: 26 mar. 2022.

JILLSON, Elisa; HASTY, Andy. **Six steps toward more secure cloud computing**. 2020. Disponível em: <https://www.ftc.gov/business-guidance/blog/2020/06/six-steps-toward-more-secure-cloud-computing>. Acesso em: 21 abr. 2022.

MELL, Peter; GRANCE, Timothy. **The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology**. Special Publication 800-145. Estados Unidos: NIST, 2011. 3 p.

MICROSOFT. **O que é um contêiner?** 2022. Disponível em: <https://azure.microsoft.com/pt-br/overview/what-is-a-container/#resources>. Acesso em: 25 abr. 2022.

NETAPP. **What are containers?** 2022. Disponível em: <https://www.netapp.com/devops-solutions/what-are-containers/>. Acesso em: 25 abr. 2022.

ORTEGA, Luis. **The IBM 360/91**. Disponível em: <http://www.columbia.edu/cu/computinghistory/36091.html>. Acesso em: 03 nov. 2021.

OS AUTORES DO KUBERNETES. **Componentes do Kubernetes**. 2022. Disponível em: <https://kubernetes.io/pt-br/docs/concepts/overview/components/>. Acesso em: 05 jun. 2022.

RED HAT, INC. **O que é YAML?** 2021. Disponível em: <https://www.redhat.com/pt-br/topics/automation/what-is-yaml>. Acesso em: 05 jun. 2022.

REINSEL, David; GANTZ, John; RYDNING, John. **Data age 2025: the evolution of data to life-critical the evolution of data to life-critical: don't focus on big data; focus on the data that's big**. 2017. Disponível em: <https://www.import.io/>. Acesso em: 27 maio 2022.

SCARFONE, Karen; SOUPPAYA, Murugiah; HOFFMAN, Paul. **Guide to Security for Full Virtualization Technologies: Recommendations of the National Institute of Standards and Technology**. Special Publication 800-125. Estados Unidos: NIST, 2011. B-1 p.

SILL, A. et. al. **NIST Cloud Computing Standards Roadmap**. Special Publication 500-291. Version 2. Estados Unidos: NIST, 2013. 102 p.

SMARTBEAR SOFTWARE. **What are Microservices?** 2022. Disponível em: <https://smartbear.com/solutions/microservices/>. Acesso em: 01 maio 2022.

STACK247. **Azure: on premises vs iaas vs paas vs saas. On Premises vs IaaS vs PaaS vs SaaS**. 2015. Disponível em: <https://stack247.wordpress.com/2015/05/21/azure-on-premises-vs-iaas-vs-paas-vs-saas/>. Acesso em: 07 nov. 2021.

TELE.SINTESE (ed.). **Google investe US\$ 140 milhões em data center no Chile**. 2018. Disponível em: <https://www.telesintese.com.br/google-invete-us-140-milhoes-em-data-center-no-chile/>. Acesso em: 24 abr. 2022.

THE KUBERNETES AUTHORS. **Implementation details**. 2022. Disponível em: <https://Kubernetes.io/docs/reference/setup-tools/kubeadm/implementation-details/>. Acesso em: 18 abr. 2022.

TORRES, Gabriel. **Arquitetura de redes**. 2022. Disponível em: <https://ibm-learning.udemy.com/course/redes-modulo-1/learn/lecture/21134696?start=0#overview>. Acesso em: 17 abr. 2022.