INSTITUTO FEDERAL DE SANTA CATARINA

GABRIEL DE SOUZA

Proposta de nuvem privada multicâmpus para o IFSC usando orquestração de contêineres

São José - SC

dezembro/2018

RESUMO

O amplo acesso à tecnologia impulsiona o movimento de transformação digital das empresas e, nesse sentido, muitas delas surgem a partir da tecnologia ou têm forte dependência da mesma. Diante disso, e respaldado pelo modelo de computação em nuvem, os *Data Centers* estão passando por profundas transformações e as demandas e necessidades postas às Tecnologias da Informação e Comunicação (TIC) cada vez mais requerem maior agilidade, efetividade e disponibilidade. Nesse ambiente surgem inúmeras tecnologias e abordagens que propõem soluções para esses novos desafios, em especial as tecnologias de virtualização de recursos de TIC, que implementam atualmente o conceito de *Data Center* definido por *software*, no qual todos os recursos (processamento, armazenamento, rede e outros) são virtuais e com tolerância à falhas e resiliência baseadas em *software*. Outra tecnologia que aborda uma melhor gestão dos serviços são os contêineres aliada aos orquestradores de contêineres. Com a migração da infraestrutura de um modelo tradicional para um novo modelo de provimento dos serviços com base nos contêineres e desacoplada do *hardware*, alguns benefícios são a utilização de infraestrutura geograficamente distribuída e, por conseguinte, alta disponibilidade. Desta forma, propõe-se a criação de uma nuvem privada multicâmpus para o Instituto Federal de Santa Catarina (IFSC) utilizando orquestração de contêineres.

Palavras-chave: Computação em Nuvem. Orquestração de Contêiner. Nuvem Privada.

ABSTRACT

This is the english abstract.

Keywords: Cloud computing. Container Orchestration. Private Cloud.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CRUSH *Controlled Replication Under Scalable Hashing* 26

MDS *Metadata Server* 26

IP *Internet Protocol* 31

IaaS *Infrastructure as a Service* 32

PaaS *Platform as a Service* 32

SaaS *Software as a Service* 32

CaaS *Container as a Service* 32

FaaS *Function as a Service* 32

XaaS *Anything as Service* 32

SUMÁRIO

1 INTRODUÇÃO

O crescente acesso às tecnologias, impulsionado pela difusão da internet e pelos avanços tecnológicos têm tornado as TICs cada vez mais estratégicas para as empresas (ITU, 2017). Sondergaard (2013) afirmou que toda empresa seria uma empresa de tecnologia, dada a importância, dependência e diferencial que esta agrega aos principais negócios das empresas. Com o objetivo de responder a crescente demanda e a velocidade das mudanças, novas tecnologias são criadas e com isso muda-se a forma de gerir os ambientes de TIC. Novas formas de criar, suportar e entregar serviços, em sua maioria *Web*, trazem uma série de desafios para os *Data Centers*, de forma que as infraestruturas tradicionais não conseguem atender o ritmo de escala adequado (BARROSO; RANGANATHAN, 2010). Diante disso, a popularização dos conceitos de virtualização de infraestrutura, computação em *cluster/grid* e principalmente o modelo de computação em nuvem mostram-se aliados importantes das empresas no que tange à agilidade das operações de TIC.

Desde o ano 2000, após a idealização do programa Governo Eletrônico[1] (substituído em 2016 pela Estratégia de Governança Digital[2]), uma série de iniciativas surgiram com o objetivo de melhorar a atuação da TIC do Governo Brasileiro. Inicialmente, foi diagnosticada a ausência de padrões de interoperabilidade, compatibilidade e desempenho nas áreas de infraestrutura e serviços do governo (BRASIL, 2002). Posteriormente, foram criados comitês[3] e grupos que geraram documentos normativos, estratégicos e de orientação para melhorar a prestação dos serviços (BRASIL, 2015). Nesse sentido, pode-se destacar a criação do documento "Padrões de Interoperabilidade de Governo Eletrônico - ePING"[4], que define um "*conjunto de premissas, políticas e especificações técnicas que regulamentam a utilização da TIC na interoperabilidade do Governo Federal* [...]", bem como do "Guia Livre – Referência de Migração para Software Livre", contendo "*dicas para a elaboração de planos de migração, relatos de experiências bem sucedidas no Governo Federal*"[5].

Em 2006 foi publicado o "Guia de Estruturação e Administração do Ambiente de Cluster e Grid", que informa sobre as novas concepções tecnológicas e alerta sobre a falta de padrões abertos, a utilização de *hardware* especializados e as relações de dependência tecnológica criadas com fornecedores (BRASIL, 2006). A partir do cenário retratado é colocada a "[...] *necessidade de busca por alternativas computacionais inovadoras e interoperáveis, com independência de fornecedor, economicamente sustentáveis e que fomentem o desenvolvimento de novas tecnologias*" (BRASIL, 2006).

As características e as diretrizes apresentadas para uma nova abordagem de infraestrutura no Guia em 2006 podem também ser encontradas nos princípios chaves dos *clusters* do Google: "*Confiabilidade de software; Uso de replicação para melhor taxa de transferência de solicitações e disponibilidade; Preço/desempenho supera o desempenho de pico; O uso de PCs de commodity reduz o custo de computação*" (BARROSO; DEAN; HOLZLE, 2003). Todos esses princípios estão alinhados com o conceito atual de computação em nuvem, ou seja, a computação em nuvem se coloca como um termo novo para conceitos já conhecidos. Prova disso são as ofertas de computação em nuvem pelas empresas públicas de prestação

[1] Governo Eletrônico (eGOV): <https://www.governodigital.gov.br/EGD/historico-1/historico>
[2] Estratégia de Governança Digital (EGD): <https://www.governodigital.gov.br/EGD/o-que-e>
[3] Comitês Técnicos instituídos, no âmbito do Comitê Executivo do Governo Eletrônico por meio do Decreto não numerado, de 29 de Outubro de 2003: <https://www.governodigital.gov.br/documentos-e-arquivos/legislacao/Decreto_29_11_2003.pdf>
[4] Padrões de Interoperabilidade de Governo Eletrônico - ePING: <http://eping.governoeletronico.gov.br/>
[5] Guia Livre – Referência de Migração para Software Livre: <https://www.governodigital.gov.br/transformacao/cidadania/software-livre/guia-livre>

de serviço em TIC como Dataprev[6], Serpro[7] e RNP[8]. A atual nuvem da Serpro (2018), por exemplo, começou em 2013 baseada nas mesmas referências e características apresentadas no "Guia de Estruturação e Administração do Ambiente de Cluster e Grid" (MAZONI, 2013).

Com o objetivo de regulamentar e priorizar a contratação de nuvens híbridas pelos órgãos do governo, o Ministério do Planejamento, Desenvolvimento e Gestão (MP) publicou o documento "Boas práticas, orientações e vedações para contratação de Serviços de Computação em Nuvem", destacando os benefícios de "[...] *redução de custos, elasticidade, redução da ociosidade dos recursos, agilidade na implantação de novos serviços, foco nas atividades finalísticas do negócio e uso mais inteligente da equipe de Tecnologia da Informação (TI)*" (BRASIL, 2016). O documento também restringe a contratação de salas cofres e a contratação direta de equipamentos de infraestrutura de TIC como servidores e *storages*. Além das nuvens públicas mencionadas anteriormente, o governo federal também está apostando na contratação de nuvens públicas de iniciativas privadas e fez em novembro de 2018 um pregão para a contratação de empresa especializada (integrador) para prestação de serviços de computação em nuvem[9] para pelo menos 12 órgãos públicos.

Realizando-se um resgate histórico sobre a busca pela alta disponibilidade dos serviços centralizados do IFSC, ilustrado na Figura 1, observou-se que a Diretoria de Tecnologia da Informação e Comunicação (DTIC) vinculada a Reitoria do IFSC possui desde 2008 serviços centralizados, como sistema acadêmico, *Webmail* e manutenção do Sophia[10] (SILVA, 2011). A partir de 2012, com a criação do Fundo de TI do IFSC e com o aumento do número de câmpus (de 7 em 2008 para mais de 20 em 2012) foram adquiridos equipamentos de *Data Center* e agregados mais serviços como, por exemplo, servidor de arquivos e impressão (IFSC, 2013).

No Plano Diretor de Tecnologia da Informação (PDTI) de 2013 aparece pela primeira vez a preocupação com a indisponibilidade dos serviços em caso de problemas no *Data Center* da reitoria, observada na necessidade N.11 que trata da "*Implantação de solução para replicação dos dados Data Center em um outro local na grande Florianópolis como medida de contingência*" e define como meta a implantação da solução até dezembro de 2013 (IFSC, 2013). O PDTI 2014-2015 informa que a necessidade N.11 não foi iniciada e adiciona a necessidade N.22 de "*Replicação dos dados da reitoria no Data Center da Educação a Distância (EaD)*" com observações sobre a necessidade da aquisição de licenças específicas e as limitações impostas pela solução adotada (IFSC, 2014). O PDTI 2016-2017 classifica a necessidade N.22 como "em execução", mas declara como "à definir" [*sic*] e adiciona a necessidade N.8 de "*Replicar 25 máquinas virtuais do Data Center da reitoria para o Data Center do Centro de Referência em Formação e Educação a Distância (CERFEAD)*" (IFSC, 2016).

Com a publicação da Portaria MP/STI nº 20, de 14 de junho de 2016, que "*Dispõe sobre orientações para a contratação de soluções de TIC no âmbito da Administração Pública Federal direta, autárquica e fundacional[...]*", são identificadas novas abordagens no PDTI da instituição. A necessidade de replicação da infraestrutura do *Data Center* ainda aparece no PDTI 2017-2018 com a identificação N.1 do item serviços de Rede "*Disponibilidade de serviços/sistemas em caso de sinistro no Data Center da DTIC*" (IFSC, 2017). Além disso, outra necessidade é retratada com a identificação N.3 no item de infraestrutura: "*Ter infraestrutura disponível para processamento, armazenamento e backup de serviços e sistemas*". Essa demanda em especial aborda a otimização de recursos físicos, de pessoal e financeiro centralizando os serviços de rede na Grande Florianópolis e é composta por ações como "*Promover infraestrutura em nuvem pública e/ou privada*", "*aquisição de equipamentos e/ou serviços*" e estima recursos orçamentários

[6] GovCloud Dataprev: <https://servicos.dataprev.gov.br/vitrine/#/govcloud>
[7] Estaleiro Serpro: <https://estaleiro.serpro.gov.br/>
[8] Nas Nuvens RNP: <https://nasnuvens.rnp.br/>
[9] Pregão Eletrônico nº 29/2018: <https://goo.gl/JRK5DB>
[10] Sistema de Bibliotecas Integradas do IFSC: <http://biblioteca.ifsc.edu.br/>

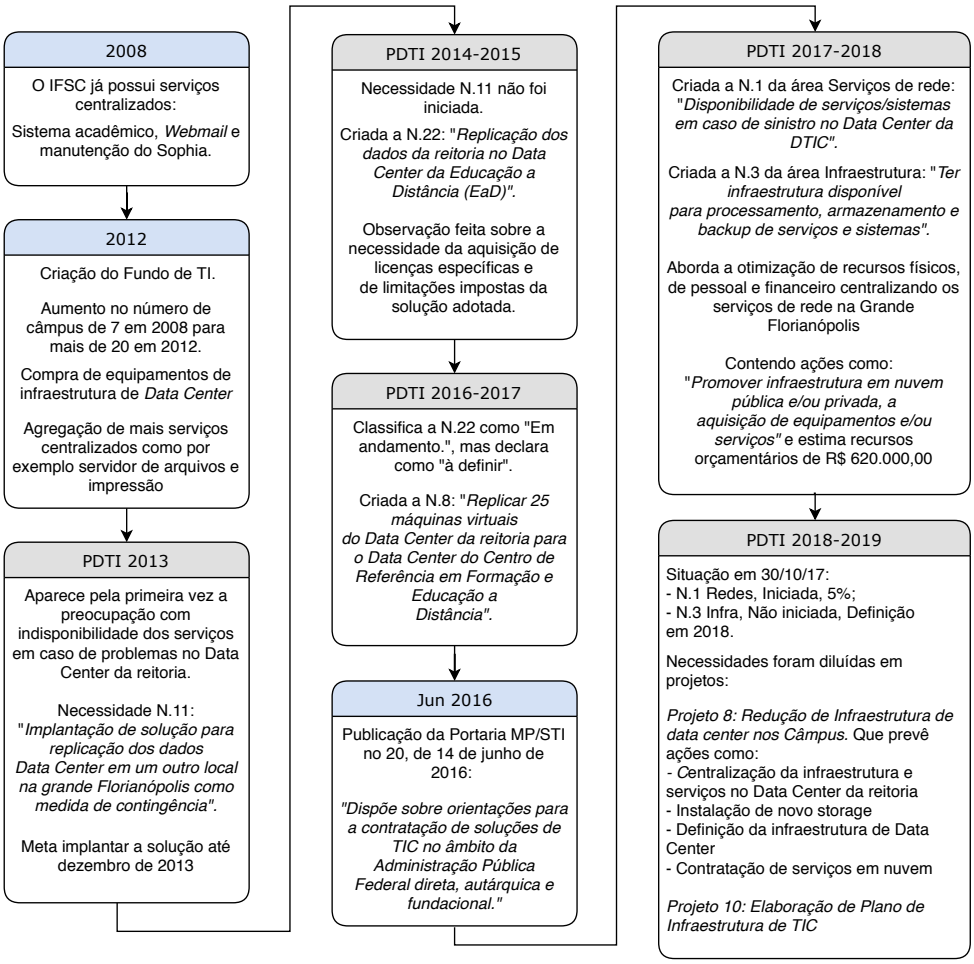


Figura 1 – Resgate histórico da centralização dos serviços de TIC do IFSC.

Fonte: Produzida pelo autor.

de R$ 620.000,00 (IFSC, 2017).

No documento seguinte, PDTI 2018-2019, foi apresentado o relatório de resultados do PDTI anterior, informando que a necessidade N.1 de redes foi iniciada e com meta alcançada de 5%. Já a necessidade N.3 de infraestrutura não havia iniciado e continha como observação: "*Definição em 2018*". O IFSC (2018a) apresenta projetos no lugar de necessidades, com destaque para os de número 8 e 10. O projeto de número 10 é intitulado "*Elaboração de Plano de Infraestrutura de TIC*". Já o projeto de número 8 é intitulado "*Redução de Infraestrutura de data center nos Câmpus*" e prevê ações como centralização da infraestrutura e serviços no *Data Center* da reitoria, instalação de novo *storage*, definição da infraestrutura de *Data Center* e contratação de serviços em nuvem (IFSC, 2018a).

Após o resgate histórico feito com base na análise dos PDTIs, representado na Figura 1, é evidente que a centralização da infraestrutura e dos serviços de TIC no IFSC vem sendo priorizada e viabilizada, porém, características essenciais como alta disponibilidade, recuperação de desastre, *site backup*, escalabilidade, automação e agilidade do *Data Center* ainda não foram implementadas. Conforme mencionado, o "Guia de Estruturação e Administração do Ambiente de Cluster e Grid", publicado em 2006, buscava reverter o quadro de dependência tecnológica do governo brasileiro propondo um novo paradigma de computação, isto é, a computação distribuída. Vale ressaltar que Brasil (2006, p. 21) faz alertas sobre restrições ou problemas relacionados à adoção de soluções baseadas na computação de grande porte:

> • Limitação financeira dos investimentos públicos e a crescente necessidade de racionalização do uso de recursos públicos em TI [...];
>
> • Dificuldade de aumentar ou diminuir a capacidade computacional de acordo com a demanda atual de cada instituição;
>
> • Dependência tecnológica e de fornecedor devido à utilização de hardware altamente especializado [...], os quais somente a empresa que comercializa o equipamento ou o software é capaz de prestar suporte, realizar manutenção ou a venda de novos componentes;
>
> • Sistemas e hardwares heterogêneos [...] muitas vezes incompatíveis entre si, devido ao emprego de padrões fechados ou arquiteturas proprietárias;
>
> • Dificuldade de integração e interoperabilidade entre os sistemas devido à abundância de padrões proprietários, segmentados e divergentes, conjugados com a falta de padrões abertos.

As restrições apresentadas acima, bem como as diferenças das características entre a computação de grande porte e a distribuída, apresentadas na Tabela 1, se mostraram relevantes motivos para a ausência de execução de necessidades recorrentes nos PDTIs da IFSC desde 2012, conforme apresentado na Figura 1.

Tabela 1 – Diferenças entre computação de grande porte e distribuída

Grande Porte	*Cluster e Grid*
- Alto custo de implantação;	- Baixo custo de implantação;
- Dependência de fornecedor único;	- Independência de fornecedores; - Facilidade de negociação;
- Utilização de *hardware* específico;	- Utilização de *hardware* comum – padrão PC;
- Alto custo de manutenção;	- Baixo custo de manutenção;
- Dificuldade de redimensionamento do ambiente;	- Facilidade de redimensionamento do ambiente;
- Utilização parcial da capacidade de processamento;	- Maximização da capacidade de processamento;
- Grande custo total de propriedade;	- Baixo custo total de propriedade;
- Tecnologia estabelecida no mercado.	- Tecnologia inovadora.

Fonte: Brasil (2006)

Com a implantação do SIG[11], Discovirtual[12] e Moodle[13] centralizados, cada vez mais a questão da disponibilidade e estabilidade da infraestrutura que dá suporte aos serviços centralizados do IFSC foi ganhando relevância. Entendendo a importância desse tema, a DTIC focou as diretrizes e projetos do Plano Estratégico de Tecnologia da Informação e Comunicação (PETIC) e do PDTIC 2018-2019 para solucionar essas necessidades. Também, foi possível verificar nos documentos a influência do documento "*Boas práticas, Orientações e Vedações*", publicado pelo governo federal em 2016 (BRASIL, 2016). Com o objetivo de apresentar a motivação para realização do presente trabalho, segue-se um extrato desses documentos institucionais nos temas relacionados aos objetivos expostos na seção 1.1.

Primeiramente, destacam-se as seguintes diretrizes do Plano Estratégico de Tecnologia da Informação e Comunicação (PETIC) do IFSC (2018b, p. 21):

> • Priorizar soluções com código aberto;
>
> • Planejar todas as contratações de TIC;
>
> • Racionalizar o uso dos recursos de TIC;
>
> • Buscar soluções inovadoras.

[11] Sistema Integrado de Gestão: <https://sig.ifsc.edu.br>
[12] Serviço de armazenamento e sincronização de arquivos: <https://discovirtual.ifsc.edu.br>
[13] Ambiente virtual de aprendizagem: <https://moodle.ifsc.edu.br/>

No PETIC 2018 foi realizada uma análise FOFA, que elenca as Forças, Oportunidades, Fraquezas e Ameaças da instituição no que no que tange as TIC e na Tabela 2 são apresentados alguns itens que justificam o presente trabalho.

Tabela 2 – Extrato da análise FOFA do PETIC 2018

Forças	**Fraquezas**
- Investimentos na Infraestrutura física (equipamentos) para suporte aos serviços do IFSC; - Autossuficiência para manutenção dos sistemas e serviços pela TI.	- Falta de redundância/replicação dos serviços centralizados.
Oportunidades	**Ameaças**
- Novas soluções de TIC disponíveis no mercado.	- Recursos financeiros contingenciados; - Dependência de fornecedores de bens e serviços.

Adaptada de IFSC (2018b)

Com base na Tabela 2, uma solução que se proponha a resolver a fraqueza elencada pelo IFSC (2018b) deve: ser encontrada com base nas novas soluções de TIC presentes no mercado; levar em consideração que a instituição fez recentes investimentos na infraestrutura física; que a equipe já possui autossuficiência para manutenção dos sistemas e serviços; e principalmente as ameaças de contingenciamento financeiro e de dependência de fornecedores. Considerando as diretrizes do planejamento estratégico, deve-se buscar soluções inovadoras, com prioridade em soluções de código aberto, de forma planejada a fim de racionalizar os recursos de TIC.

Nesse sentido destacam-se algumas soluções e iniciativas que servem como referencial, pois se enquadram nos requisitos apresentados:

- A solução de armazenamento de dados em alta disponibilidade e geograficamente distribuídos adotada pela Universidade Federal de Santa Catarina (UFSC), implementada com uma ferramenta de armazenamento definido por *software*, gerou redução de custo do *Terabyte* armazenado em aproximadamente 60% a 83% (GERONIMO; HIPOLITO, 2018);
- A solução de infraestrutura em nuvem privada adotada pela Coordenadoria de Tecnologia de Informação e Comunicação (CTIC) do Câmpus São José do IFSC, baseada em orquestração de contêineres (IFSC - SJE, 2018b). Bem como as modernizações na infraestrutura de *Data Center* com automação[14] e infraestrutura como código (IFSC - SJE, 2018a).

Corroborando com as diretrizes e estratégias do PETIC, os projetos do PDTIC buscam atender essas demandas, prevendo a capacitação dos servidores para atender a necessidade institucional e viabilizando a disseminação de conhecimento por meio de cursos, grupos de estudos e atuação por regiões. Concomitantemente, projetam o compartilhamento de conhecimento e de soluções que tragam benefício a instituição e gerem economia de recursos financeiros. Outra pilar dos projetos do PDTIC 2018-2019 é a centralização dos serviços, redução da infraestrutura de *Data Center* nos câmpus e ações de ampliação e redundância nos enlaces de Internet nos Câmpus (IFSC, 2018a).

No PDTIC 2018-2019 é citada a contratação de nuvem para suporte a infraestrutura e serviços do IFSC. Dado o perfil do histórico dos PDTIs (Figura 1), fica evidente a "substituição" da necessidade

[14] Automação com Ansible: <https://github.com/ctic-sje-ifsc/ansible>

de redundância/alta disponibilidade de *Data Center* para uma aposta no modelo de nuvem (IFSC, 2018a). Essa evolução tecnológica requer alguns estágios de maturidade, conforme Figura 2. Ao passo que, nos últimos anos a instituição fez os investimentos na infraestrutura física para suporte ao atendimento da demanda institucional.

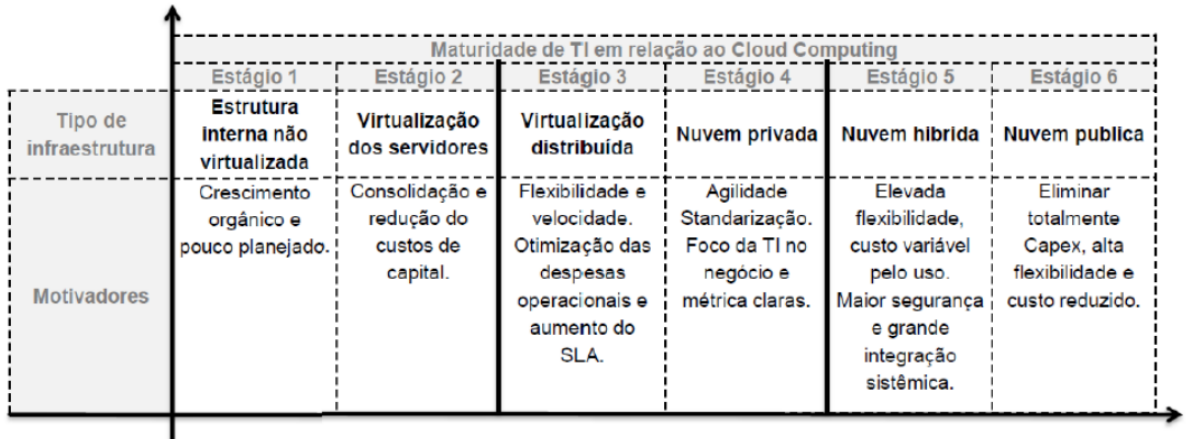


Figura 2 – Evolução tecnológica para a Nuvem.

Fonte: Junior (2014)

Dessa forma, o presente trabalho se propõe a abordar uma solução que esteja alinhada com as diretrizes e estratégias do PETIC, com os projetos do PDTIC, com as políticas do governo federal e com as tendências tecnológicas nacionais e internacionais, visando atender as novas tecnologias e metodologias de *Data Center* amplamente aceitas e utilizadas.

1.1 Objetivo

Validar, por meio de prova de conceito, a viabilidade da criação de uma nuvem privada multicâmpus (geograficamente distribuída) utilizando o *hardware* legado adquirido pela instituição (CTICs e DTIC).

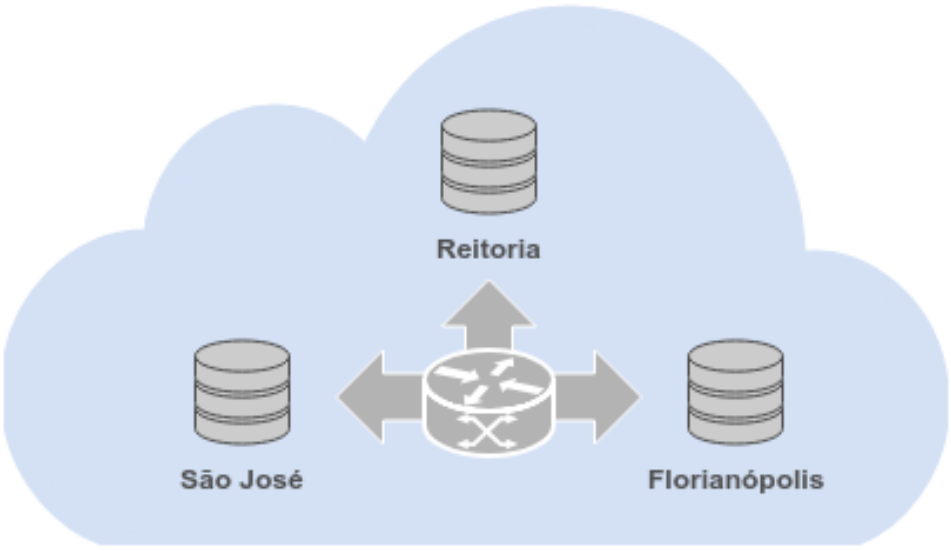


Figura 3 – Modelo da estrutura proposta como nuvem privada multicâmpus

Fonte: Produzida pelo autor.

Os objetivos específicos são:

1. Propor uma nova abordagem para resiliência dos serviços centralizados baseada em soluções definidas por *software*;
2. Gerar diretrizes estruturantes sobre uma nova abordagem de infraestrutura de TIC e enviá-las ao Comitê de Tecnologia da Informação (CTI) do IFSC, com o intenção de auxiliar no cumprimento dos objetivos do PETIC e dos projetos do PDTI.

1.2 Metodologia

A metodologia proposta para atingir os objetivos consiste na validação das seguintes etapas:

1. Avaliação da infraestrutura atual do IFSC, por meio da analise dos PDTIs, dos documentos de empenho e da consulta com aos setores de TIC.
2. Implantação, a partir do fornecimento da infraestrutura necessária pela CTIC do câmpus São José, do câmpus Florianópolis e da DTIC, de um sistema de armazenamento distribuído entre os campus da grande Florianópolis por meio de uma ferramenta de armazenamento definido por *software*.
3. Implantação, a partir do fornecimento da infraestrutura necessária pela CTIC do câmpus São José, do câmpus Florianópolis e da DTIC, de uma infraestrutura de computação em nuvem com a ferramenta de orquestração de contêineres, utilizando para armazenamento persistente o armazenamento distribuído implantado.
4. Implantação de dois serviços críticos da instituição como o Moodle centralizado e o Disco Virtual nessa nova estrutura, bem como todas as configurações necessárias para prover balanceamento de carga, alta disponibilidade e escalabilidade automática.
5. A partir do conhecimento adquirido na avaliação da infraestrutura atual, na implantação das estruturas, na validação dos serviços portados, e ainda, nas políticas do governo federal, nas tendências tecnológicas nacionais e internacionais escrever diretrizes estruturantes, que serão abordadas nos seguintes temas:
    - implementação de uma nuvem privada multicâmpus utilizando a infraestrutura atual;
    - migração dos serviços atuais para a nova estrutura;
    - atualização e manutenção da capacidade computacional e de armazenamento;
    - contratação de serviços de nuvens públicas de TIC para o IFSC.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Sistemas distribuídos

Segundo Steen e Tanenbaum (2017), sistema distribuído é um conjunto de computadores independentes que se apresentam aos seus usuários como um sistema único e coerente. Semelhantemente, Coulouris et al. (2013) define como um sistema no qual os componentes de *hardware* ou *sofware*, localizados em computadores interligados em rede, se comunicam e coordenam suas ações apenas enviando mensagens entre si. Diante das definições, tem-se que o objetivo principal dos sistemas distribuídos é utilizar os recursos de TIC de forma compartilhada e distribuída a fim de aumentar a confiabilidade e propensão a alterações de demanda.

Existem algumas formas de agrupar os recursos para usufrui-los de forma compartilhada, sendo que os principais são: computação em *cluster*, computação em *grid* e computação em nuvem. A computação em *cluster* é definida como um conjunto de computadores que, interconectados por redes de alta velocidade, trabalham mutuamente para execução de uma mesma tarefa, apresentado-se como um único recurso de alto desempenho (STEEN; TANENBAUM, 2017). Já a computação em *grid* utiliza recursos de computadores heterogêneos e geograficamente distribuídos, geralmente de diferentes organizações para atingir uma escala muito grande para solução de problemas específicos (COULOURIS et al., 2013). Segundo Vaquero et al. (2009), computação em nuvem é um grande conjunto de recursos virtualizados facilmente utilizável e acessível. A computação em nuvem é detalhada na seção 2.5.

2.1.1 Escalabilidade

Uma das características dos sistemas distribuídos é a escalabilidade, que é conceituada como a competência de um sistema modificar a sua capacidade para atender mudanças na carga de trabalho, bem como a susceptibilidade a sua ampliação (BONDI, 2000). Coulouris et al. (2013) afirma que um sistema é dito escalável quando permanece eficiente mesmo com mudanças significativas no número de recursos ou usuários.

Segundo Michael et al. (2007), há duas abordagens para o escalonamento de *hardware* nas infraestruturas de data center, as quais são a escalabilidade vertical e a escalabilidade horizontal:

- **Escalabilidade vertical**: aumento de recurso de um único nó em um sistema, como adição de mais processadores, memória ou armazenamento.
- **Escalabilidade horizontal**: aumento de recurso baseado no incremento de nós de computação ou armazenamento.

2.1.2 Tolerância a falhas

Sistemas de computação, em especial os de grande porte, falham. A tolerância a falhas é a habilidade dos sistemas em assimilar falhas de *software* e *hardware* sem interferir na disponibilidade dos serviços aos usuários, ou seja, mantê-los funcionando de maneira aceitável mesmo na presença de falhas (DUBROVA, 2013). Segundo Steen e Tanenbaum (2017), o controle de falhas é muito importante na criação de sistemas confiáveis. Falhas parciais irão ocorrer, e nesse sentido os sistemas distribuídos são aliados na implantação de sistemas tolerantes a falhas, pois a confiabilidade é obtida através de técnicas de redundância reduzindo ou eliminando pontos únicos de falhas (COULOURIS et al., 2013).

2.1.3 Sistemas abertos

Segundo Steen e Tanenbaum (2017) e Coulouris et al. (2013), uma das características mais importantes em sistemas distribuídos é ele ser aberto, podendo ser estendido, integrado e reimplementado de diferentes formas. Um sistema aberto tem como primeira característica a disponibilização das principais interfaces de *software* e suas especificações, comumente definindo os serviços por meio de uma *Interface Definition Language* (IDL). Esses sistemas geralmente são projetados a partir de especificações de padrões e protocolos abertos disponibilizados como *Request for Comments* (RFC)s[1] ou no *World Wide Web Consortium* (W3C)[2]. Dessa forma, o modelo aberto torna os sistemas mais interoperáveis, portáveis e extensíveis (STEEN; TANENBAUM, 2017).

Ultimamente, vem ganhando força o modelo de desenvolvimento global de projetos de código aberto que, a partir de sua notável popularidade e interesse, são incubados em fundações compostas por inúmeras empresas e pessoas interessadas (comunidades) que fornecem um ambiente independente e democrático para esses projetos. Muitos desses projetos, diante da maciça adoção e popularização, inclusive pelas grandes empresas de tecnologia mundial, tornaram-se "padrões" em determinados nichos (Opensource, 2018). Podem-se citar as seguintes fundações: OpenStack Foundation[3], The Linux Foundation[4], *Cloud Native Computing Foundation* (CNCF)[5], Ceph Foundation[6] e Apache Software Foundation[7]. Um exemplo de projeto que evidencia esse fato é a ferramenta para orquestração de contêineres Kubernetes, apresentado na subseção 2.4.1.

2.1.4 Sistemas distribuídos baseados na arquitetura *Web*

A comunicação é de suma importância para os sistemas distribuídos, Steen e Tanenbaum (2017) afirmam que a comunicação está no coração de todo sistema distribuído. Pode-se afirmar que por intermédio das trocas de mensagens é que são implementadas características essenciais dos sistemas distribuídos. Existem várias formas de implementar as trocas de mensagens, utilizando protocolos de níveis mais baixos ou protocolos de alto nível. Para os sistemas baseados na *Web* o protocolo *HyperText Transfer Protocol* (HTTP) é o padrão para troca de mensagens.

O HTTP é um protocolo de comunicação do tipo cliente-servidor que define como um cliente interage com os servidores *Web* (COULOURIS et al., 2013). Este protocolo classifica as mensagens envolvidas em uma troca de requisição-resposta, seus métodos, argumentos, resultados e as regras para representá-los (empacotá-los) na comunicação.

Para classificar e acessar recursos dos sistemas *Web*, é utilizada a *Uniform Resource Identifier* (URI) que é uma sequência de caracteres que se refere a um recurso. A URI, pode ser de dois tipos: *Uniform Resource Name* (URN), que refere-se a identificação única de um recurso, independente da sua localização, como um *International Standard Book Number* (ISBN) de um livro ; ou *Uniform Resource Locator* (URL), que identifica onde um recurso pode ser encontrado na internet, contendo a seguinte estrutura, conforme Mozilla (2018):

```
esquema ou protocolo://domínio:porta/caminho/recurso?query_string#fragmento
```

A eficácia da utilização de protocolos simples, base para inúmeros serviços e aplicações, é provada pelo crescimento da *Web* nas ultimas duas décadas (COULOURIS et al., 2013). O protocolo de requisição-

[1] RFCs: <https://www.ietf.org/standards/rfcs/>
[2] World Wide Web Consortium (W3C): <https://www.w3.org/standards/>
[3] OpenStack Foundation: <https://www.openstack.org/foundation/>
[4] The Linux Foundation: <https://www.linuxfoundation.org/projects/>
[5] CNCF: <https://www.cncf.io/projects/>
[6] Ceph Foundation: <https://ceph.com/foundation/>
[7] Apache Software Foundation: <https://www.apache.org/foundation/>

resposta HTTP, em especial pela sua simplicidade e versatilidade, foi adotado amplamente, inclusive servindo de base para o modelo *Representational State Transfer* (REST), adotado nas implementações de Serviços *Web*. REST é um estilo de arquitetura para projetar aplicações distribuídas fracamente acopladas bastante utilizado no desenvolvimento de Serviços *Web* (RESTFULAPI, 2018). No REST, a abstração chave de informação é um recurso, portanto, um recurso é qualquer informação que possa ser nomeada e cada um possui uma URI.

Com esses princípios, foram adotadas *Application Programming Interface* (API)s baseadas na *Web*, como a REST API, com sintaxes diretas e semântica definida para interações de aplicações distribuídas (FIELDING, 2000). Um formato comumente usado para estruturação e representação do conteúdo dos recursos na REST API é *JavaScript Object Notation* (JSON)[8]. O JSON é construído como uma coleção de pares chave-valor, sendo realizado como objeto, registro, estrutura, entre outros, e em alguns casos, pode ser apresentado como uma lista ordenada de valores. Um fator importante sobre linguagens como o JSON ou o YAML[9](que segue os mesmos princípios) é que elas são facilmente legíveis para humanos e também para as máquinas as analisarem e gerarem (JSON, 2018).

2.2 Data Center definido por *software*

Historicamente, um dos fatores que possibilitou a evolução dos *Data Centers* para que eles pudessem cada vez mais atender as crescentes demandas foram as diferentes tecnologias de virtualização (COULOURIS et al., 2013). A virtualização, de forma genérica, permite a criação de ambientes simulados e o compartilhamento de recursos físicos (*hardware*) de forma dedicada. A técnica de virtualização pode ser usada em diferentes contextos, para o compartilhamento de recursos como armazenamento, processamento e rede, entre outros (BACHIEGA et al., 2018). Sem dúvida, a virtualização de servidores baseada na criação de máquinas virtuais (*Virtual Machine* (VM)) é a forma mais conhecida de virtualização, a qual consiste na divisão, por meio de *software* (hipervisor), de um único sistema de *hardware* físico em ambientes virtuais simulados e isolados, disponibilizados como máquinas virtuais, que podem atender a diferentes funções (RedHat, 2018d).

Já a virtualização de rede, conhecida como *Software-defined Networking* (SDN), é uma abordagem que inova no projetar, implementar e gerenciar redes, desacoplando o plano de dados e o plano de controle. Os benefícios das implementações de SDN são: melhor eficiência na configuração, maior flexibilidade nos novos projetos das redes, desempenho superior e maior eficiência (XIA et al., 2015). Outra tecnologia de virtualização muito importante no contexto do *Software-defined Data Center* (SDDC) é o Armazenamento definido por *software* (*Software-defined Storage* (SDS)), abordado em separado no subseção 2.2.1.

Esse novo paradigma de infraestrutura de TIC, com abordagens que implementam os conceitos de *Software Defined* (SD), garante melhor eficiência e agilidade na gestão dos recursos de TIC (BARROSO; HöLZLE; RANGANATHAN, 2018). E, ao mesmo tempo, surgem abordagens que definem a infraestrutura como código (*Infrastructure As Code* (IaC)), possibilitando implantações e manutenções mais dinâmicas e ágeis (KLEIN, 2018).

O *Data Center* definido por *software* é a consolidação da virtualização, culminando em um "*Data Center* virtual". Nesse sentido, é percebido um alinhamento com tendências de sistemas distribuídos baseados na arquitetura *Web*, apoiados por protocolos simples e em comunicação via API. Outro fator importante são as técnicas de automação flexibilizando e conferindo agilidade à infraestrutura, bem como independência do *hardware*. Tudo isso possibilita a adoção de metodologias ágeis, cultura DevOps[10] para

[8] Introducing JSON: <https://www.json.org/>
[9] YAML: <http://yaml.org/>
[10] DevOps: <https://pt.wikipedia.org/wiki/DevOps>

as TICs e o modelo de infraestrutura em nuvem, abordado no seção 2.5 (CHERIAN, 2018).

2.2.1 Armazenamento definido por *software*

O armazenamento definido por software é uma arquitetura de armazenamento de dados distribuída com a característica principal de ser independente de *hardware*. Diferente dos sistemas tradicionais de armazenamento de dados em rede (*Network-Attached Storage* (NAS)) ou das redes de área de armazenamento (*Storage Area Network* (SAN)), o SDS é projetado para ser executado em qualquer arquitetura x86[11] eliminando, assim, a dependência de *software* e *hardware* proprietários (HEWLETT PACKARD ENTERPRISE, 2018). Por isso, o SDS é mais ágil e econômico do que as abordagens tradicionais e possibilita que a capacidade de armazenamento seja baseada em escalabilidade horizontal, ampliando rapidamente de forma mais econômica e flexível.

2.2.2 Ceph

O Ceph é uma plataforma de armazenamento definido por *software* de código aberto, desenvolvido para fornecer armazenamento de objetos, blocos e arquivos nativamente em um sistema unificado (GERONIMO; HIPOLITO, 2018). Este *software* é projetado para eliminação de pontos únicos de falha (*Single Point of Failure* (SPOF)), com base em uma plataforma distribuída, autogerenciada e autorrecuperável (UEHARA et al., 2018). A solução tem se apresentado como uma substituta a altura para soluções tradicionais de armazenamento (ZHANG; GADDAM; CHRONOPOULOS, 2015).

Um *cluster* Ceph é composto por Monitores, Gerentes, Ceph *Object Storage Daemon* (OSD) e *Metadata Server* (MDS). Segundo Ceph (2018), podemos descrever os componentes do Ceph como:

- **Monitores**: Um Ceph *Monitor* (ceph-mon) é responsável por guardar mapas do estado do *cluster*, contendo o mapa do monitor, o mapa do gerenciador, o mapa OSD e o mapa *Controlled Replication Under Scalable Hashing* (CRUSH), além de ser responsável por gerenciar a autenticação entre *daemons* e clientes. É indicado o uso de pelo menos três monitores para redundância e alta disponibilidade.
- **Gerentes**: Ceph *Manager* (ceph-mgr) tem a função de acompanhar as métricas de tempo de execução e o estado atual do *cluster* do Ceph, incluindo as métricas de desempenho atuais, a utilização do armazenamento e o carga do sistema. Para obtenção de alta disponibilidade geralmente são necessários pelo menos dois gerentes.
- **Ceph OSDs**: o Ceph OSD (ceph-osd) armazena os dados de fato, implementa a replicação, recuperação e rebalanceamento de dados e fornece algumas informações de monitoramento para os Monitores e Gerentes do Ceph. Pelo menos 3 Ceph OSDs são normalmente necessários para redundância e alta disponibilidade.
- **MDSs**: Um Ceph MDS (ceph-mds) armazena metadados[12] em nome do Ceph *Filesystem* (ou seja, o armazenamento em bloco e em objeto não usam MDS). Ele permite que os usuários do sistema de arquivos POSIX possam executar comandos básicos (como *ls*, *find*, etc.).

2.3 Contêineres

Contêineres são uma forma de virtualização em nível de sistema operacional, que visa promover um ambiente de execução de aplicações compartilhando o *kernel* do Sistema Operacional (SO) hospedeiro

[11] Arquitetura x86: <https://pt.wikipedia.org/wiki/X86>
[12] Metadados: permissões, propriedades, datas etc.

(BACHIEGA et al., 2018). Embora o termo tenha sido cunhado nos últimos anos, a ideia do que conhecemos atualmente por contêineres surgiu em 2000 com o *FreeBSD jails*[13] e avançou no decorrer dos anos, principalmente no isolamentos dos processos e espaços de usuário (RedHat, 2018c).

Segundo Bachiega et al. (2018), ao contrário das virtualizações tradicionais, os contêineres não utilizam as camadas de emulação de *hardware* (hipervisores) e de SOs convidados para executarem. Consequentemente, conforme podemos observar na Figura 4, o *overhead* diminui em comparação com as máquinas virtuais. Já que os contêineres compartilham o SO hospedeiro, são necessários mecanismos de proteção para o isolamento de processos, sendo que os dois principais são o *namespaces* e o *cgroups*, ambos presentes no *kernel* do Linux (TRINDADE, 2018).

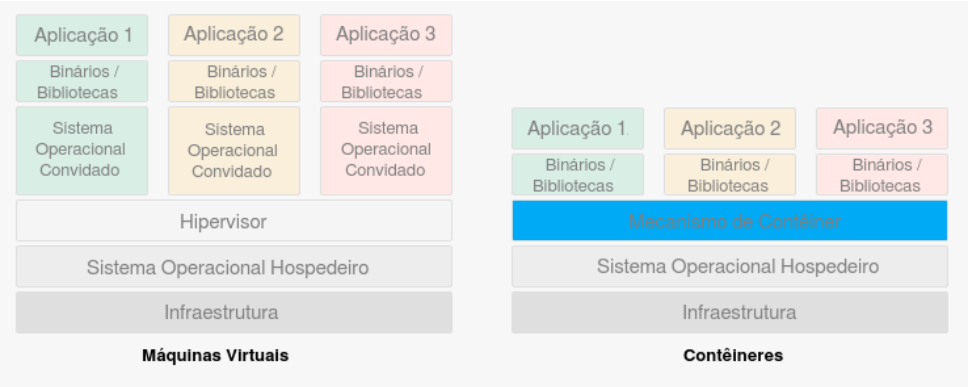


Figura 4 – Virtualização baseada em hipervisor e em contêiner.

Fonte: Adaptado de Google (2018)

Os contêineres são usados para empacotar aplicações e suas dependências de forma independente e desacoplada do ambiente de destino, permitindo a implantação e manutenção de maneira fácil e consistente (GOOGLE, 2018). Segundo Kubernetes (2018h), podemos resumir o benefício dos contêineres nos seguintes itens:

Criação e implantação ágil de aplicativos: Maior facilidade e eficiência na criação de imagens de contêiner em comparação ao uso de imagens da VM;

Desenvolvimento, integração e implementação contínuos: Proporciona criação e implantação confiável e frequente de imagens de contêiner com reversões rápidas e fáceis (devido à imutabilidade da imagem);

Separação de interesses Dev e Ops: Cria imagens de contêiner de aplicativo no momento da compilação/liberação, ao invés de no tempo de implementação, desacoplando assim os aplicativos da infraestrutura;

Observabilidade: Não apenas visualiza informações e métricas no nível do sistema operacional, mas também a integridade do aplicativo e outros sinais;

Consistência ambiental em desenvolvimento, testes e produção: executa o mesmo em um *laptop* e na nuvem;

Portabilidade de distribuição de nuvem e SO: funciona no Ubuntu, RHEL, CoreOS, no Google, no Google Kubernetes Engine e em qualquer outro lugar;

Gerenciamento centrado em aplicativo: aumenta o nível de abstração de executar um sistema operacional em *hardware* virtual para executar um aplicativo em um sistema operacional usando recursos lógicos;

Microsserviços fracamente acoplados, distribuídos, elásticos e liberados: os aplicativos são divididos em partes menores e independentes e podem ser implantados e gerenciados dinamicamente - não uma pilha monolítica executada em uma grande máquina de propósito único;

Isolamento de recursos: desempenho de aplicativo previsível;

Utilização de recursos: alta eficiência e densidade.

[13] FreeBSD jails: <https://www.freebsd.org/doc/handbook/jails.html>

Existem inúmeras tecnologias e ferramentas de contêineres, podem-se citar como principais: Docker[14], LXC[15] e OpenVz[16]. Em 2015 foi criado o projeto *Open Container Initiative (OCI)*[17] com o propósito de criar padrões de mercado abertos em torno de formatos de contêiner e tempo de execução. A tecnologia Docker, que é de código aberto, é atualmente o projeto e o método mais famoso para implantar e gerenciar contêineres e é o método escolhido para o desenvolvimento desse trabalho.

2.3.1 Docker

Conforme apresentado no seção 2.3, o Docker é a ferramenta mais popular no mundo dos contêineres. Essa hegemonia se deve principalmente ao fato do Docker ter iniciado o movimento que popularizou o uso facilitado dos contêineres, revolucionando os mecanismos para administração de contêineres de maneira rápida, automatizada e simplificada (BACHIEGA et al., 2018).

Segundo Turnbull (2016), a plataforma do Docker é composto pelos seguintes componentes:

- ***Docker Engine***: é um processo em segundo plano que faz o trabalho de executar os contêineres, manipulando as imagens, rede e armazenamento;
- **Imagens**: são os blocos de construção do mundo Docker. Pode-se considerar as imagens como o "código fonte" dos contêineres Docker. São altamente portáveis, criadas em camadas e versionadas. As imagens são construídas a partir de "receitas", em um formato chamado Dockerfile[18];
- **Registros (registry)**: as imagens construídas são armazenadas nos registros, que podem ser públicos ou privados. Os registros públicos como o Docker Hub[19] possuem milhares de imagens prontas que podem ser utilizadas;
- **Contêineres Docker**: pode ser descrito como a imagem em execução, com todos os requisitos para suporte da aplicação em execução.

2.4 Orquestração de contêineres

A popularização dos contêineres, alicerçada nas novas metodologias[20] de desenvolvimento de softwares-como-serviços e na arquitetura de microsserviços[21] (caracterizada pela concepção de que cada aplicação execute em contêineres separados), resulta no aumento exponencial da quantidade de contêineres (RedHat, 2018a).

Os desafios para gerenciar essa nova arquitetura baseada em contêineres em escala são inúmeros: clusterização[22], alta disponibilidade, escalonamento, gerenciamento de integridade, rede, armazenamento persistente, entre outros. Podem-se destacar os registros, a rede, o armazenamento, a segurança e a telemetria como os pilares que devem ser atendidos pela orquestração de contêiner (Figura 5). Diante disso, foi necessário a criação de ferramentas que pudessem orquestrar os contêineres e tratar desses desafios, como o Kubernetes (BURNS et al., 2016).

[14] Docker: <https://www.docker.com/>
[15] *Linux Containers* (LXC): <https://linuxcontainers.org/>
[16] OpenVz: <https://openvz.org/>
[17] Open Container Initiative (OCI): <https://www.opencontainers.org/about>
[18] *Dockerfile reference*: <https://docs.docker.com/engine/reference/builder/>
[19] Docker Hub: <https://hub.docker.com/>
[20] *The Twelve-Factor App* : <https://12factor.net/pt_br/>
[21] Microservice architecture pattern: <https://microservices.io/patterns/microservices.html>
[22] Implementação em múltiplos servidores de maneira distribuída.

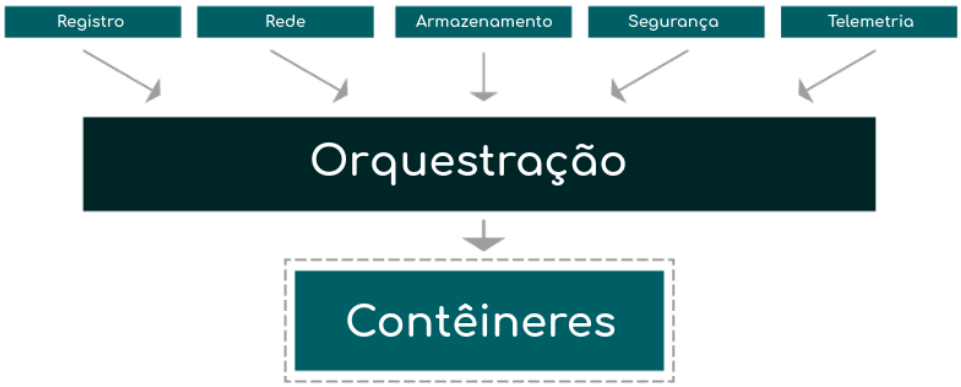


Figura 5 – Pilares da orquestração de contêineres.

Fonte: Adaptado de RedHat (2018b)

2.4.1 Kubernetes

Kubernetes (k8s) é um sistema de código aberto para automatizar a implantação, o dimensionamento e o gerenciamento de aplicações em contêineres em múltiplos servidores (KUBERNETES, 2018e). O k8s foi construído a partir da experiência do Google[23] na execução das suas aplicações de grande escala em produção, exclusivamente com contêineres, por mais de 15 anos (VERMA et al., 2015; BURNS et al., 2016). Em 2015, na versão 1.0, o Google cedeu o controle sobre o Kubernetes para a CNCF, que é administrada pela *The Linux Foundation*[24].

Segundo RedHat (2018b), o Kubernetes possibilita:

> - Orquestrar containers em vários *hosts*;
> - Aproveitar melhor o *hardware* para maximizar os recursos necessários na execução das aplicações corporativas;
> - Controlar e automatizar as implantações e atualizações de aplicações;
> - Montar e adicionar armazenamento para executar aplicações com monitoração de estado;
> - Escalar rapidamente as aplicações em contêineres e recursos relacionados;
> - Gerenciar serviços de forma declarativa, garantindo que as aplicações sejam executadas sempre da mesma maneira como foram implantadas;
> - Verificar a integridade e autorrecuperação das aplicações com posicionamento, reinício, replicação e escalonamento automáticos.

O k8s é implantado no modelo de *cluster* e é composto por vários componentes, subdivididos em dois tipos: componentes *Master* e componentes *node*. Segundo Kubernetes (2018a), para configurar um *clusters* k8s de alta disponibilidade são necessárias réplicas, em pelo menos 3 *hosts*, dos componentes do plano de controle. O conjunto dos componentes *master* fornece o plano de controle do *cluster* e faz toda a gestão das decisões globais do mesmo, detectando e respondendo aos eventos (KUBERNETES, 2018c). Kubernetes (2018c) lista os seguintes tipos de componentes *master*:

- **kube-apiserver**: Componente que expõe API do kubernetes. É o *front-end* do plano de controle;
- **etcd**: Armazenamento de chave-valor altamente consistente e disponível, usado como armazenamento de apoio do Kubernetes para todos os dados de *cluster*;

[23] Google: <https://www.google.com/about/>
[24] *As Kubernetes Hits 1.0, Google Donates Technology To Newly Formed Cloud Native Computing Foundation*: <https://goo.gl/81jjFW>

- **kube-scheduler**: faz o escalonamento de um recém criado *pod* para ser executado em um determinado *node*. Alguns fatores que influenciam no escalonamento são: os requisitos de recursos individuais e coletivos, restrições de *hardware*/*software*/políticas, especificações de afinidade e antiafinidade e localidade dos dados;
- **kube-controller-manager**: Componente que executa os seguintes *controllers*:
  - *Node Controller*: responsável por notificar e responder quando houver problemas de queda nos *nodes*;
  - *Replication Controller*: mantém a quantidade correta de *pods* que foram requisitadas;
  - *Endpoints Controller*: faz a associação entre os *pods* e *services*;
  - *Service Account and Token Controller*: responsável por criar contas padrões e *tokens* de acesso a API a novos *namespaces*.

Já os componentes do tipo *node*, conforme Kubernetes (2018c), executam em cada nó mantendo os *pods* em execução e provendo o ambiente de execução do k8s. Esses componentes são:

- **kubelet**: responsável por garantir que os contêineres sejam saudáveis e estejam em execução, com base nas *PodSpecs*;
- **kube-proxy**: habilita a abstração de serviço do k8s mantendo as regras de rede do *host* e por meio de encaminhamento de conexões;
- ***Container Runtime***: *software* responsável pela execução dos contêineres.

A Figura 6 representa de forma ilustrativa os componentes do *cluster* Kubernetes:

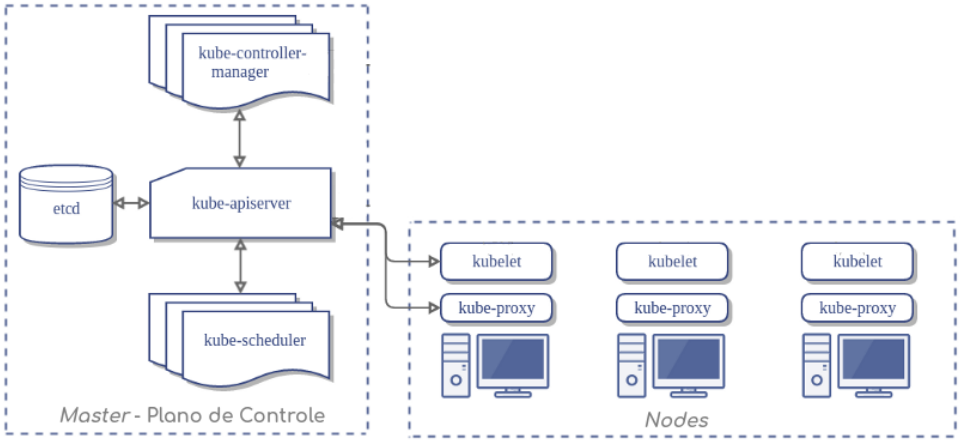


Figura 6 – Componentes do Kubernetes

Fonte: Adaptado de Kubernetes (2018c)

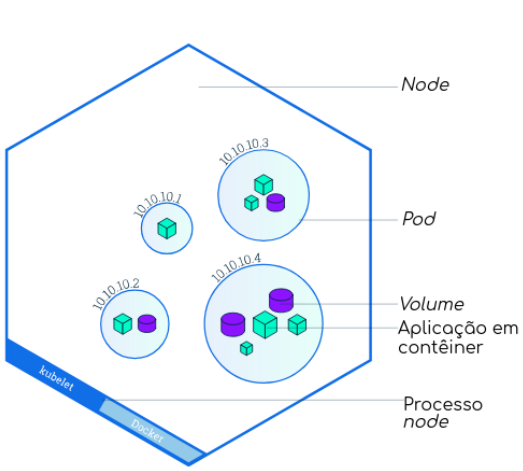
Segundo Kubernetes (2018b), a a estrutura fundamental do Kubernetes é a API REST. O servidor de API manipula todas as as operações e comunicações entre componentes e comandos de usuário externos como chamadas de API REST. A API do Kubernetes é baseada em recursos (RESTful) fornecida via HTTP[25]. Portanto, no Kubernetes tudo é tratado como um objeto da API e possui uma entrada correspondente nela. Com o objetivo de facilitar o desenvolvimento e garantir que os mecanismos sejam

[25] *API OVERVIEW:* <https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.13/>

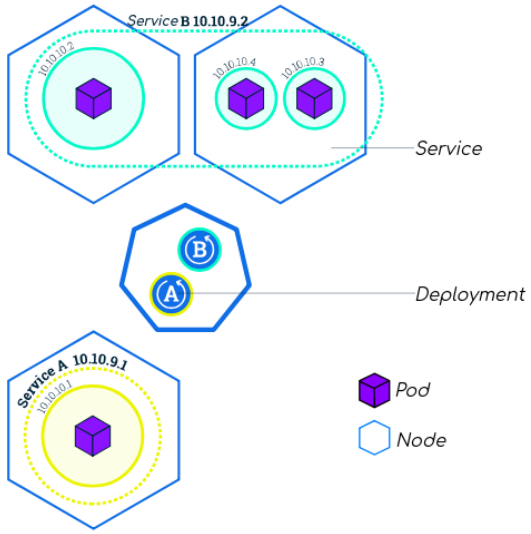
implementados e funcionem de forma consistente em diferentes casos de uso, a API é aberta e padronizada em um documento de convenção[26].

O k8s implementa o conceito de objetos, representados na API, para abstrair todas as representações do estado do sistema: aplicativos e cargas de trabalho, recursos de rede e armazenamento e outras informações sobre o *cluster* (KUBERNETES, 2018d). De acordo com Kubernetes (2018f), os objetos básicos, ilustrados na Figura 7, são descritos como:

- ***Pod***: o menor objeto implementável no modelo de objetos do k8s. Representa um processo em execução por intermédio de um ou mais contêineres, um *Internet Protocol* (IP) de rede e as opções de execução dos contêineres.
- ***Service***: é uma abstração que define grupos lógicos de *pods* que executam a mesma função como uma entidade descrevendo como acessar esses aplicativos por portas ou balanceamento através da rede IP;
- ***Volume***: diretório acessível aos contêineres em um *pod* contendo dados. Resolve "problemas" de manutenção e compartilhamento de dados relativos às características de efemeridade e de agrupamento de contêineres por *pod*;
- ***Namespace***: implementação de "*cluster* virtuais", ou seja, execução de multiplos *clusters* no mesmo *cluster* físico;
- ***Ingress***: gerencia o acesso externo aos serviços em um cluster, geralmente HTTP.



(a) Representação do *pod* e componentes básicos.



(b) Representação básica do componente *service*

Figura 7 – Objetos básicos kubernetes

Fonte: Adaptado de Kubernetes (2018g)

Além dos objetos básicos, existem abstrações em níveis superiores, chamados Controladores. Segundo Kubernetes (2018d), podem ser descritos como:

- ***ReplicaSet***: responsável por garantir que o número especificado de *pods* esteja sempre em execução;

[26] *API Conventions*: <https://github.com/kubernetes/community/blob/master/contributors/devel/api-conventions.md>

- ***Deployment***: implementa e atualiza os *pods* e *ReplicaSets* para um estado desejado;
- ***StatefulSets***: é o objeto da API usado para gerenciar aplicativos com estado.
- ***DaemonSet***: garante a execução de um *pod* por *node*.

2.5 Computação em nuvem

A computação em nuvem, conforme Badger et al. (2012), é definida como um modelo de computação que visa a abstração dos recursos de infraestrutura computacional local (redes, servidores, armazenamento, aplicativos e serviços) em agrupamentos de recursos virtuais. Trata-se de um novo paradigma em computação, no qual há entrega rápida e sob demanda de recursos de TIC e aplicativos pela rede, considerando assim, tudo como serviço, desde infraestrutura física ou virtual até *software* (VAQUERO et al., 2009).

As principais características da computação em nuvem são: alocação de recursos sob demanda, amplo acesso a rede, agrupamento de recursos, rápida elasticidade e serviço medido (BADGER et al., 2012). Os princípios agrupados no modelo de computação em nuvem não são novos pois remetem aos fundamentos dos sistemas distribuídos, principalmente dos serviços *Web* possibilitados pelo aumento da velocidade das conexões e o avanço nas tecnologias de virtualização dos *Data Centers*, entre outros fatores (TAURION, 2009).

Badger et al. (2012) também definem os modelos de serviços e modelos de implantação, abordados na subseção 2.5.1 e subseção 2.5.2, respectivamente.

2.5.1 Modelos de serviços

Na computação em nuvem, podem ser oferecidas diferentes formas de acesso aos recursos, com diferentes responsabilidades e especificidades. Segundo Badger et al. (2012), existem três modelos de serviço: Infraestrutura como Serviço ou *Infrastructure as a Service* (IaaS), Plataforma como um Serviço ou *Platform as a Service* (PaaS) e *Software* como Serviço ou *Software as a Service* (SaaS). Os autores os definem como:

- ***IaaS***: na infraestrutura de nuvem como serviço o consumidor têm acesso a recursos de computação tradicionais como processamento, armazenamento e redes. Com isso pode fazer a execução de *softwares* arbitrários, escolhendo o SO, a plataforma de aplicações e as aplicações, conforme a Figura 8. O usuário não administra ou gerencia esses recursos físicos e de provisionamento.
- ***PaaS***: a plataforma de nuvem como serviço consiste no modelo em que são fornecidas como serviços plataformas para criação de aplicações, por meio de linguagens e ferramentas, ou seja, uma infraestrutura de apoio para todo o ciclo de desenvolvimento de uma aplicação. O gerenciamento feito pelo usuário é restrito às implementações e configurações das aplicações disponíveis, conforme ilustrado na Figura 8.
- ***SaaS***: no modelo de nuvem de *software* como um serviço os recursos são acessíveis por interfaces *Web*, como os navegadores, e alguns serviços são as ferramentas de escritório, *email*, serviços de armazenamento, entre outros. Conforme representado na Figura 8, no modelo SaaS o usuário não gerencia nenhum item da infraestrutura de suporte.

Atualmente são encontrados nas referências comerciais outros modelos, como *Container as a Service* (CaaS) e *Function as a Service* (FaaS), que acabaram cunhando o termo *Anything as Service* (XaaS),

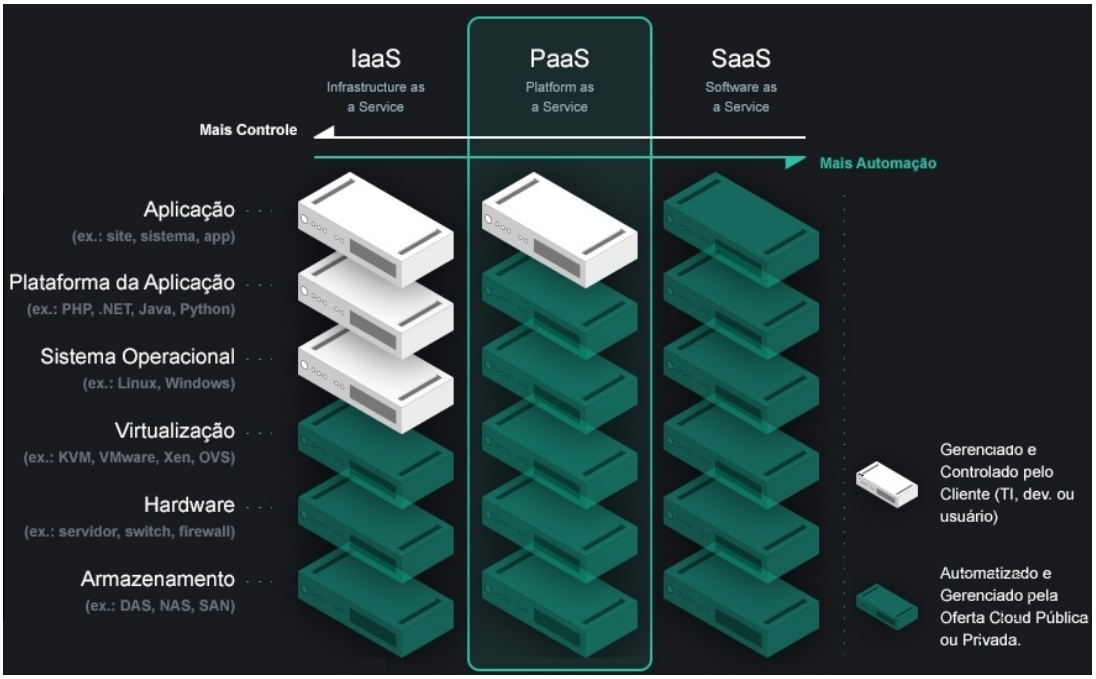


Figura 8 – Modelos de serviços

Fonte: Adaptado de EVEO (2018)

que define "qualquer coisa como serviço". Porém, esses outros modelos são recentes e não existem definições formalizadas, podendo ser entendidos como partes dos modelos já abordados.

2.5.2 Modelo de implantação

Badger et al. (2012) caracteriza a implantação da computação em nuvem em diferentes modelos, sendo eles: Nuvem Pública, Privada, Híbrida e Comunitária.

- **Nuvem Privada**: é quando uma empresa possui uma infraestrutura em nuvem e disponibiliza os recursos exclusivamente aos usuários da organização nesse modelo;
- **Nuvem Pública**: esse modelo implementa o provisionamento compartilhado da infraestrutura de nuvem por diferentes organizações e públicos. Geralmente os provedores de nuvem fazem a tarifação baseada no uso ou no perfil de uso das organizações clientes;
- **Nuvem Híbrida**: quando uma organização faz o uso de mais de um modelo de implantação para provisionamento dos seus recursos, como por exemplo possuindo aplicações que são executadas em nuvem privada mas podem fazer o transbordo de carga para uma Nuvem Pública diz-se que implementa uma Nuvem Híbrida;
- **Nuvem Comunitária**: semelhantemente ao modelo de Nuvem Privada, os recursos são provisionados para uso exclusivo de uma comunidade específica que compartilha algum objetivo, como por exemplo entre universidades.

2.6 Nuvens privadas locais

Dentro do modelo de implantação de nuvem privada existem diferentes escopos. O principal deles é o modelo de nuvens privadas locais, no qual a infraestrutura de nuvem é implantada no *Data Center* da

própria instituição (BADGER et al., 2012). Segundo Hsu, Chang e Hsu (2017) em uma nuvem privada local podemos destacar as seguintes vantagens: segurança, privacidade, personalização, acessibilidade e custo:

Segurança: É bastante aumentado, pois o acesso pode ser limitado a redes privadas. Conformidade com regulamentos ou restrições específicas de criptografia ou armazenamento é mais fácil de ser cumprida.

Privacidade: Frequentemente, códigos, softwares ou dados privados e proprietários são desenvolvidos e testados em sistemas de larga escala. Usar esse software em uma Nuvem Privada Local, onde os componentes de rede, bem como a Tradução de Endereço de Rede (NAT), podem ser controlados, facilitam um nível de proteção de informações proprietárias e limitam as chances de exfiltração;

Customizabilidade: O hardware pode ser personalizado para os requisitos exatos da aplicação: CPUs, coprocessadores, aceleradores e as redes podem ser adaptadas à carga de trabalho. Da mesma forma, o software pode ser personalizado através do uso de componentes *Open Source*;

Acessibilidade: As soluções de nuvem local permitem alta largura de banda e baixa latência, bem como colocação física e acesso ao hardware, se necessário;

Custo: Colocação de vários *clusters* na mesma arquitetura de nuvem ajuda a reduzir custos. Em escala e quando o uso da arquitetura personalizada é necessário, é possível obter economias de custo importantes quando comparado ao custo de armazenamento e uso de nuvens públicas para experimentos de longa duração.

REFERÊNCIAS

BACHIEGA, N. G. et al. Container-based performance evaluation: A survey and challenges. In: *2018 IEEE International Conference on Cloud Engineering (IC2E)*. [S.l.: s.n.], 2018. p. 398–403. Citado 3 vezes nas páginas 25, 27 e 28.

BADGER, L. et al. *Cloud Computing Synopsis and Recommendations*. 2012. Disponível em: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-146.pdf>. Acesso em: 7 dez. 2018. Citado 3 vezes nas páginas 32, 33 e 34.

BARROSO, L. A.; DEAN, J.; HOLZLE, U. Web search for a planet: The google cluster architecture. *IEEE Micro*, v. 23, n. 2, p. 22–28, March 2003. ISSN 0272-1732. Citado na página 15.

BARROSO, L. A.; HöLZLE, U.; RANGANATHAN, P. The datacenter as a computer: Designing warehouse-scale machines, third edition. *Synthesis Lectures on Computer Architecture*, v. 13, n. 3, p. i–189, 2018. Disponível em: <https://doi.org/10.2200/S00874ED3V01Y201809CAC046>. Citado na página 25.

BARROSO, L. A.; RANGANATHAN, P. Guest editors' introduction: Datacenter-scale computing. *IEEE Micro*, v. 30, p. 6–7, 08 2010. ISSN 0272-1732. Disponível em: <doi.ieeecomputersociety.org/10.1109/MM.2010.63>. Citado na página 15.

BONDI, A. B. Characteristics of scalability and their impact on performance. In: *Proceedings of the 2Nd International Workshop on Software and Performance*. New York, NY, USA: ACM, 2000. (WOSP '00), p. 195–203. Disponível em: <http://doi.acm.org/10.1145/350391.350432>. Citado na página 23.

BRASIL. Casa Civil da Presidência da República. Ministério do Planejamento, Orçamento e Gestão. *2 Anos de Governo Eletrônico - Balanço de Realizações e Desafios Futuros*. 2002. Disponível em: <https://www.governodigital.gov.br/documentos-e-arquivos/E15_90balanco_2anos_egov.pdf>. Acesso em: 10 dez. 2018. Citado na página 15.

BRASIL. Ministério do Planejamento, Desenvolvimento e Gestão. Secretaria de Tecnologia da Informação. *Boas práticas, orientações e vedações para contratação de Serviços de Computação em Nuvem*. 2016. Disponível em: <https://www.governodigital.gov.br/documentos-e-arquivos/Orientacao%20servicos%20em%20nuvem.pdf>. Acesso em: 20 nov. 2018. Citado 2 vezes nas páginas 16 e 18.

BRASIL. Ministério do Planejamento, Orçamento e Gestão. *Governo Eletrônico Brasileiro*. 2015. Disponível em: <https://www.governodigital.gov.br/EGD/historico-1/historico>. Acesso em: 23 nov. 2018. Citado na página 15.

BRASIL. Ministério do Planejamento, Orçamento e Gestão. Secretaria de Logística e Tecnologia da Informação. *Guia de Estruturação e Administração do Ambiente de Cluster e Grid*. 2006. Disponível em: <https://www.governodigital.gov.br/documentos-e-arquivos/guiacluster.pdf>. Acesso em: 20 nov. 2018. Citado 3 vezes nas páginas 15, 17 e 18.

BURNS, B. et al. Borg, omega, and kubernetes. *Queue*, ACM, New York, NY, USA, v. 14, n. 1, p. 10:70–10:93, jan. 2016. ISSN 1542-7730. Disponível em: <http://doi.acm.org/10.1145/2898442.2898444>. Citado 2 vezes nas páginas 28 e 29.

CEPH. *INTRO TO CEPH*. 2018. Disponível em: <http://docs.ceph.com/docs/master/start/intro/>. Acesso em: 4 dez. 2018. Citado na página 26.

CHERIAN, B. *What Is the Software Defined Data Center and Why Is It Important?* 2018. Disponível em: <http://allthingsd.com/20130613/what-is-the-software-defined-data-center-and-why-is-it-important/>. Acesso em: 3 dez. 2018. Citado na página 26.

COULOURIS, G. et al. *Sistemas Distribuídos: Conceitos e Projeto*. 5. ed. [S.l.]: Bookman, 2013. Citado 3 vezes nas páginas 23, 24 e 25.

DUBROVA, E. *Fault-Tolerant Design*. [S.l.]: Springer Publishing Company, Incorporated, 2013. ISBN 1461421128, 9781461421122. Citado na página 23.

EVEO. *Container Cloud OpenShift - EVEO*. 2018. Disponível em: <https://www.eveo.com.br/container-cloud/>. Acesso em: 8 dez. 2018. Citado na página 33.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado), 2000. AAI9980887. Citado na página 25.

GERONIMO, G. A.; HIPOLITO, F. Ceph – dados geo-distribuídos de baixo custo e alta disponibilidade. *XII Workshop de Tecnologia da Informação e Comunicação das Instituições Federais de Ensino Superior do Brasil (WTICIFES 2018)*, 2018. Disponível em: <https://eventos.unila.edu.br/wticifes2018/wp-content/uploads/2018/06/98158.pdf>. Acesso em: 6 dez. 2018. Citado 2 vezes nas páginas 19 e 26.

GOOGLE. *O que são contêineres e suas vantagens | Google Cloud*. 2018. Disponível em: <https://cloud.google.com/containers/>. Acesso em: 30 nov. 2018. Citado na página 27.

HEWLETT PACKARD ENTERPRISE. *O que é armazenamento definido por software?* 2018. Disponível em: <https://www.hpe.com/br/pt/what-is/software-defined-storage.html>. Acesso em: 4 dez. 2018. Citado na página 26.

HSU, H.-H.; CHANG, C.-Y.; HSU, C.-H. *Big Data Analytics for Sensor-Network Collected Intelligence*. 1st. ed. Orlando, FL, USA: Academic Press, Inc., 2017. ISBN 0128093935, 9780128093931. Citado na página 34.

IFSC - SJE. *Cluster bare metal de Kubernetes via RKE em hosts com RancherOS(ou CoreOS)*. 2018. Disponível em: <https://github.com/ctic-sje-ifsc/baremetal_rke_kubernetes>. Acesso em: 11 dez. 2018. Citado na página 19.

IFSC - SJE. *Nuvem Privada com Kubernetes*. 2018. Disponível em: <https://github.com/ctic-sje-ifsc/servicos_kubernete>. Acesso em: 11 dez. 2018. Citado na página 19.

INSTITUTO FEDERAL DE SANTA CATARINA. *Plano Diretor de Tecnologia da Informação e Comunicação (PDTI) 2013*. 2013. Disponível em: <http://dtic.ifsc.edu.br/files/pdti2013-revisao02.pdf>. Acesso em: 21 nov. 2018. Citado na página 16.

INSTITUTO FEDERAL DE SANTA CATARINA. *Plano Diretor de Tecnologia da Informação e Comunicação (PDTI) 2014-2015*. 2014. Disponível em: <http://dtic.ifsc.edu.br/files/pdti-2014-2015-1a-revisao.pdf>. Acesso em: 21 nov. 2018. Citado na página 16.

INSTITUTO FEDERAL DE SANTA CATARINA. *Plano Diretor de Tecnologia da Informação e Comunicação (PDTI) 2016-2017*. 2016. Disponível em: <http://dtic.ifsc.edu.br/files/PDTI_2016_2017.pdf>. Acesso em: 21 nov. 2018. Citado na página 16.

INSTITUTO FEDERAL DE SANTA CATARINA. *Plano Diretor de Tecnologia da Informação e Comunicação (PDTI) 2017-2018*. 2017. Disponível em: <http://dtic.ifsc.edu.br/files/consup_resolucao12_2017_pdti.pdf>. Acesso em: 21 nov. 2018. Citado 2 vezes nas páginas 16 e 17.

INSTITUTO FEDERAL DE SANTA CATARINA. *Plano Diretor de Tecnologia da Informação e Comunicação (PDTIC) 2018-2019*. 2018. Disponível em: <http://www.ifsc.edu.br/documents/23567/0/consup_resolucao03_2018_completa_pdtic.pdf>. Acesso em: 17 nov. 2018. Citado 3 vezes nas páginas 17, 19 e 20.

INSTITUTO FEDERAL DE SANTA CATARINA. *Plano Estratégico de Tecnologia da Informação e Comunicação (PETIC) 2018-2019*. 2018. Disponível em: <http://www.ifsc.edu.br/documents/23567/0/consup_resolucao02_2018_petic.pdf>. Acesso em: 17 nov. 2018. Citado 2 vezes nas páginas 18 e 19.

ITU, I. T. U. *ICT Facts and Figures 2017*. 2017. Disponível em: <https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2017.pdf>. Acesso em: 23 nov. 2018. Citado na página 15.

JSON. *Introducing JSON*. 2018. Disponível em: <https://www.json.org/>. Acesso em: 3 dez. 2018. Citado na página 25.

JUNIOR, J. G. *Estudo de Caso: SERPRO - Computação em Nuvem - Utilização de Redes Privadas, Públicas e Híbridas no âmbito de Governo*. 2014. Disponível em: <http://www.secop2014.pr.gov.br/arquivos/File/arq_palestras/3006/1300_JoseGomes_SECOP2014_ComputacaoemNuvem.pdf>. Acesso em: 10 dez. 2018. Citado na página 20.

KLEIN, J. *Infrastructure as Code: Moving Beyond DevOps and Agile*. 2018. Disponível em: <https://insights.sei.cmu.edu/sei_blog/2018/06/infrastructure-as-code-moving-beyond-devops-and-agile.html>. Acesso em: 11 dez. 2018. Citado na página 25.

KUBERNETES. *Creating Highly Available Clusters with kubeadm*. 2018. Disponível em: <https://kubernetes.io/docs/setup/independent/high-availability/>. Acesso em: 28 nov. 2018. Citado na página 29.

KUBERNETES. *Kubernetes API Overview*. 2018. Disponível em: <https://kubernetes.io/docs/reference/using-api/api-overview/>. Acesso em: 7 dez. 2018. Citado na página 30.

KUBERNETES. *Kubernetes Components*. 2018. Disponível em: <https://kubernetes.io/docs/concepts/overview/components/>. Acesso em: 28 nov. 2018. Citado 2 vezes nas páginas 29 e 30.

KUBERNETES. *Kubernetes Concepts*. 2018. Disponível em: <https://kubernetes.io/docs/concepts/>. Acesso em: 7 dez. 2018. Citado na página 31.

KUBERNETES. *Production-Grade Container Orchestration - Kubernetes*. 2018. Disponível em: <https://kubernetes.io/>. Acesso em: 28 nov. 2018. Citado na página 29.

KUBERNETES. *Standardized Glossary - Kubernetes*. 2018. Disponível em: <https://kubernetes.io/docs/reference/glossary>. Acesso em: 7 dez. 2018. Citado na página 31.

KUBERNETES. *Tutorials - Kubernetes*. 2018. Disponível em: <https://kubernetes.io/docs/tutorials/>. Acesso em: 7 dez. 2018. Citado na página 31.

KUBERNETES. *What is Kubernetes?* 2018. Disponível em: <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>. Acesso em: 28 nov. 2018. Citado na página 27.

MAZONI, M. V. F. *Nuvem do Serpro será 100% em código aberto*. 2013. Disponível em: <http://convergenciadigital.uol.com.br/cgi/cgilua.exe/sys/start.htm?infoid=33659&sid=11#.UYke1UlGapF>. Acesso em: 23 nov. 2018. Citado na página 16.

MICHAEL, M. et al. Scale-up x scale-out: A case study using nutch/lucene. In: *2007 IEEE International Parallel and Distributed Processing Symposium*. [S.l.: s.n.], 2007. p. 1–8. ISSN 1530-2075. Citado na página 23.

Mozilla. *Identificando recursos na web*. 2018. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Basico_sobre_HTTP/Identifying_resources_on_the_Web>. Acesso em: 1 dez. 2018. Citado na página 24.

Opensource. *Open source organizations*. 2018. Disponível em: <https://opensource.com/resources/organizations>. Acesso em: 11 dez. 2018. Citado na página 24.

RedHat. *Introdução ao DevOps*. 2018. Disponível em: <https://www.redhat.com/pt-br/topics/devops>. Acesso em: 11 dez. 2018. Citado na página 28.

RedHat. *O que é o Kubernetes?* 2018. Disponível em: <https://www.redhat.com/pt-br/topics/containers/what-is-kubernetes>. Acesso em: 30 nov. 2018. Citado na página 29.

RedHat. *O que é um container Linux?* 2018. Disponível em: <https://www.redhat.com/pt-br/topics/containers/whats-a-linux-container>. Acesso em: 30 nov. 2018. Citado na página 27.

RedHat. *Qual é a diferença entre cloud e virtualização?* 2018. Disponível em: <https://www.redhat.com/pt-br/topics/cloud-computing/cloud-vs-virtualization>. Acesso em: 30 nov. 2018. Citado na página 25.

RESTFULAPI. *REST Architectural Constraints*. 2018. Disponível em: <https://restfulapi.net/rest-architectural-constraints/>. Acesso em: 3 dez. 2018. Citado na página 25.

SERPRO. *Os destaques da indústria open-source*. 2018. Disponível em: <https://estaleiro.serpro.gov.br/#technology>. Acesso em: 23 nov. 2018. Citado na página 16.

SILVA, J. G. da. *A importância do Fundo de TI do IFSC*. 2011. Disponível em: <https://jesuegraciliano.wordpress.com/reflexoes/a-importancia-do-fundo-de-ti-do-ifsc/>. Acesso em: 20 nov. 2018. Citado na página 16.

SONDERGAARD, P. *Everyone is a Technology Company*. 2013. Disponível em: <https://blogs.gartner.com/peter-sondergaard/everyone-is-a-technology-company/>. Acesso em: 17 nov. 2018. Citado na página 15.

STEEN, M. van; TANENBAUM, A. S. *Distributed Systems*. 3. ed. [S.l.]: CreateSpace Independent Publishing Platform, 2017. Citado 2 vezes nas páginas 23 e 24.

TAURION, C. *Cloud Computing: Computação em nuvem: Transformando o mundo da Tecnologia da Informação*. Rio de Janeiro: Brasport, 2009. Citado na página 32.

TRINDADE, L. V. P. *Análise do Desempenho da Virtualização Leve para Ambientes com Edge Computing baseada em NFV*. Dissertação (Mestrado) — COPPE/UFRJ, Rio de Janeiro, 2 2018. Disponível em: <http://www.gta.ufrj.br/ftp/gta/TechReports/Leon18.pdf>. Citado na página 27.

TURNBULL, J. *The Docker Book Containerization is the new virtualization*. [S.l.]: Amazon Digital Services LLC, 2016. Citado na página 28.

UEHARA, K. et al. Feasibility study of location-conscious multi-site erasure-coded ceph storage for disaster recovery. In: *2018 IEEE International Conference on Cloud Engineering (IC2E)*. [S.l.: s.n.], 2018. p. 204–210. Citado na página 26.

VAQUERO, L. M. et al. A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, dez. 2009. Disponível em: <http://doi.acm.org/10.1145/1496091.1496100>. Citado 2 vezes nas páginas 23 e 32.

VERMA, A. et al. Large-scale cluster management at Google with Borg. In: *Proceedings of the European Conference on Computer Systems (EuroSys)*. Bordeaux, France: [s.n.], 2015. Citado na página 29.

XIA, W. et al. A survey on software-defined networking. *IEEE Communications Surveys Tutorials*, v. 17, n. 1, p. 27–51, Firstquarter 2015. ISSN 1553-877X. Citado na página 25.

ZHANG, X.; GADDAM, S.; CHRONOPOULOS, A. T. Ceph distributed file system benchmarks on an openstack cloud. In: *2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. [S.l.: s.n.], 2015. p. 113–120. Citado na página 26.