Mathias Hillesheim

Chatbot para monitoramento e controle de orquestrador de nuvem de contêineres

São José - SC

junho/2018

Mathias Hillesheim

Chatbot para monitoramento e controle de orquestrador de nuvem de contêineres

Monografia apresentada à Coordenação de Engenharia de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma Bacharel em Engenharia de Telecomunicações.

Instituto Federal de Santa Catarina – IFSC

Câmpus São José

Engenharia de Telecomunicações

Orientador: Ederson Torresini

São José - SC

junho/2018

Mathias Hillesheim Chatbot para monitoramento e controle de orquestrador de nuvem de contêineres/ Mathias Hillesheim. – São José - SC, junho/2018-
Orientador: Ederson Torresini

Monografia (Graduação) – Instituto Federal de Santa Catarina – IFSC
Câmpus São José
Engenharia de Telecomunicações, junho/2018.

1. Chatbot. 2. Contêineres. 3. API. I. Ederson Torresini. II. Instituto Federal de Santa Catarina. III. Câmpus São José. IV. Chatbot para monitoramento e controle de orquestrador de nuvem de contêineres

Mathias Hillesheim

Chatbot para monitoramento e controle de orquestrador de nuvem de contêineres

Monografia apresentada à Coordenação de Engenharia de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma Bacharel em Engenharia de Telecomunicações.

Trabalho aprovado. São José - SC, ?? de junho de 2018:

Prof. Ederson Torresini, Me.
Orientador

Prof. Marcelo Maia Sobral, Dr.
IFSC

Prof. Marcos Moecke, Dr.
IFSC

São José - SC
junho/2018

Dedico este trabalho a minha, e a todas as mães do mundo, que dedicam as suas vidas para ensinar aos seus filhos que nunca deve-se ter medo de sonhar.

Agradecimentos

Agradeço, primeiramente, a minha mãe, sem a qual eu nunca chegaria até onde cheguei.

Agradeço também a cada um dos professores pelos quais eu passei. Por mais difícil que seja enxergar, eu sei que no fundo todos contribuíram para o que eu sou hoje.

Agradeço, em especial, ao Ederson Torresini, que além de orientador, foi um grande amigo durante todos os momentos do desenvolvimento desse trabalho.

E por fim, agradeço a você, por doar um pouco do seu tempo lendo-o, já que eu me esforcei tanto para escrevê-lo, um dia.

"Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)

Resumo

A internet revolucionou a computação e a comunicação como nada antes já visto. Seu grande crescimento trouxe um avanço tecnológico sem precedentes, mas também trouxe a tona diversos paradigmas estruturais no decorrer das últimas quatro décadas. Quando o uso de máquinas reais se mostrou ineficiente quanto ao aproveitamento dos seus recursos, surgiram as máquinas virtuais gerenciadas pelos *Hypervisors*. Com o grande *overhead* associado ao uso de sistemas virtuais, uma nova era se iniciou com a popularização de contêineres e o uso de complexos orquestradores.

Atualmente, mesmo o uso de orquestradores de *clusters* de contêineres não entrega um gerenciamento simples e práticos dos seus serviços quanto estes são numerosos. Motivado por essa problemática que é enfrentado atualmente pela Coordenadoria de Tecnologia da Informação e Comunicação (CTIC) do Instituto Federal de Santa Catarina (IFSC) câmpus São José, esse trabalho busca a implementação de um *software* dedicado a simplificar a interação do usuário com o orquestrador de contêineres, criando uma comunicação entre a *Application Programming Interface* (API) fornecida pelo orquestrador e um *ChatBot* que pode ser disponibilizado nas ferramentas de *chat* da equipe.

Palavras-chave: Chatbot. Contêineres. API.

Abstract

The internet has revolutionized computing and communication as nothing ever seen before. Its growth has brought unprecedented technological advances, but it has also brought several structural paradigms over the past four decades. When the use of real machines proved to be inefficient regarding the use of its resources, the virtual machines managed by Hypervisors appeared. With the large overhead associated with the use of virtual systems, a new era began with the popularization of containers and the use of complex orchestrators to manage it.

Currently, even the use of container clusters does not deliver simple and practical management of their services as they are numerous. Motivated by this problematic that is currently faced by the CTIC of the *IFSC câmpus São José*, this work seeks the implementation of a software dedicated to simplify the user interaction with the container orchestrator, creating a communication between the API provided by the orchestrator and a *ChatBot* that can be delivered on the *chat* application of the team.

Keywords: Chatbot. Container. API.

Lista de ilustrações

Lista de listagem de códigos

Lista de abreviaturas e siglas

NLP *Natural-language processing* 33

CPD Centro de processamento de dados 24

Sumário

1 Introdução

A internet revolucionou a computação e a comunicação como nada antes já visto. Desde o seu início, ainda quando denominada ARPANET, até os dias atuais, se desenvolveu de maneira gigantesca, tornando-se unanimemente a maior ferramenta de disseminação de informação, e um meio de colaboração e interação entre indivíduos e seus computadores independente de localização geográfica.(LEINER et al., 2009). Segundo (SALEH; SIMMONS, 2011), o tráfego da Internet continua exibindo um crescimento exponencial. Em 1995, a população mundial com acesso a internet era equivalente a 0.4%, mas em dezembro de 2017 esse número já havia crescido para 54.4% (GROUP, 2018).

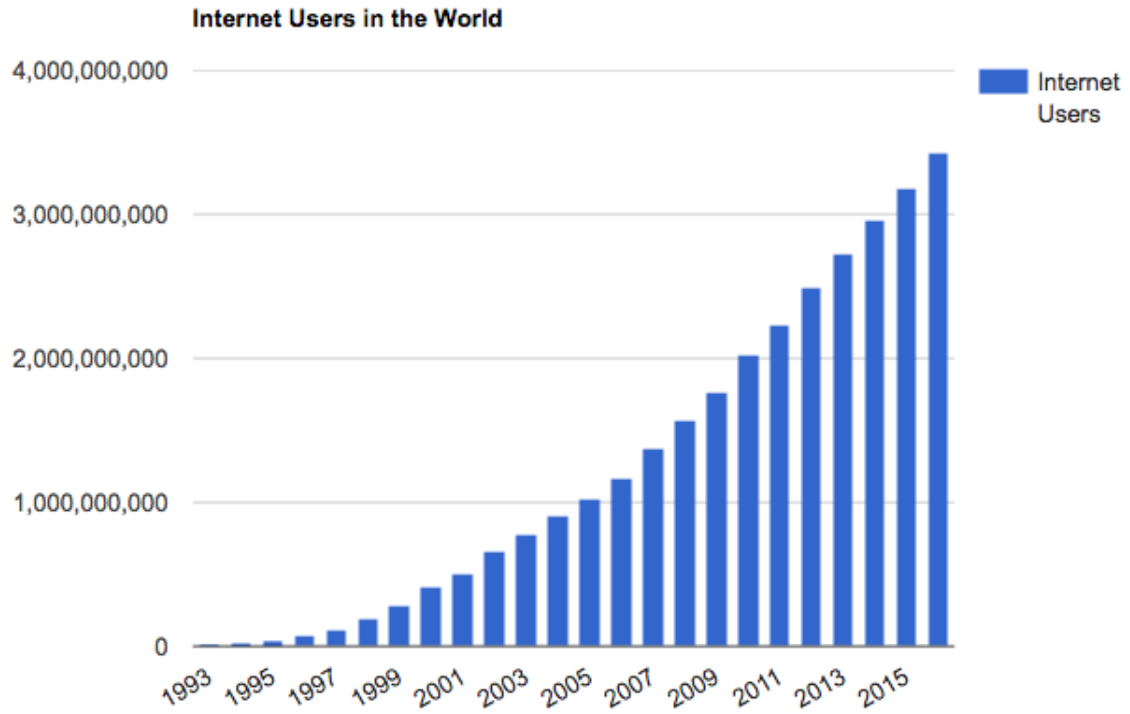

Figura 1 – Usuários da internet pelo mundo ao longo dos anos, segundo (STATS, 2018).

O crescimento da internet trouxe também o grande crescimento do uso de computadores e equipamentos responsáveis por manter a internet funcionando, aumentando a complexidade da sua estrutura, e também o seu custo. Não demorou muito para que a prática da virtualização de computadores se popularizasse, buscando reduzir a grande demanda de *hardware* da internet, fazendo também uso de *Network Functions Virtualization* (NFV), um conceito de arquitetura de rede que usa técnicas de virtualização para virtualizar classes inteiras de nós da rede em blocos que podem se conectar para criar um serviço de comunicação (HAN et al., 2015).

O termo máquina virtual, citado pela primeira vez na década de 1960, se resume a "uma abstração em *software* que emule as características de uma máquina real". Uma maneira simplificada de entender a virtualização é considerá-la como uma fatia da pilha de *hardware* e/ou *software* de um computador (ROSENBLUM, 2004). Máquinas Virtuais permitem que usuários criem, copiem, salvem, leiam, modifiquem e façam inúmeras outras ações com o estado de uma máquina, com todas as facilidades da manipulação de um simples arquivo. Isso gera um grande valor agregado no que diz respeito a usuários e principalmente administradores de sistemas. Sua flexibilidade permite um processo muito simplificado de migração, onde máquinas reais são gradualmente substituídas por seus equivalentes virtuais. Por consequência de todas as facilidades que apresenta, tem-se observado um grande crescimento na adoção da virtualização no cenário atual da internet (GARFINKEL; ROSENBLUM, 2005).

O crescimento da internet citado anteriormente causou o adensamento de Centro de processamento de dados (CPD)s. Dessa forma, servidores em rede se tornaram uma das aplicações mais importantes em qualquer grande sistema de computadores. Principalmente os servidores *World Wide Web* (WWW), que recorrentemente são acessados por milhares ou mesmo milhões de usuários simultaneamente. O antigo conceito de virtualização foi resgatado para se atingir maior eficiência, e o caminho para contêineres foi natural da tecnologia nesse sentido.

Contêiner é um pacote executável de um pedaço de *software* onde se é permitido incluir tudo que é necessário para executá-lo: código, ferramentas de sistema, bibliotecas de sistema, configurações. Com tal ferramenta, é possível isolar a aplicação do que a cerca, ajudando a reduzir conflitos na execução de diferentes programas na mesma infraestrutura.

No desenvolvimento moderno, o uso de contêineres ajuda a implementar uma cultura de microsserviços e faz com que as aplicações cada vez menos sejam monolíticas (NEWMAN, 2015). Uma prática muito comum atualmente é a de desenvolver uma aplicação que consiste em vários componentes de um sistema livremente acoplados em contêineres. Para que tal estrutura funcione, tais contêineres precisam trabalhar em conjunto. Ao trabalho de gerenciamento geral, persistência dos dados, criação de redes virtuais e endereços de recursos e *endpoints*, e descoberta de serviços de um grupo de contêineres se dá o nome de orquestração de contêineres. Existem diversas aplicações disponíveis que oferecem o serviço de orquestração, como por exemplo o Apache Mesos e o Google Kubernetes.

Tais aplicações orquestradoras oferecem diversas ferramentas para criação e administração dos contêineres que são executados dentro de seus sistemas. Porém, quando o número de aplicações gerenciadas é muito grande, a rotina de manutenção de tais serviços fica deveras burocrática. Alguns problemas recorrentes dentro de uma infraestrutura podem gerar uma repetição de determinadas ações, fazendo que mesmo todas as facilidades entregues por um orquestrador não sejam capazes de deixar o mantenimento dos serviços

simples o suficiente.

Alguns orquestradores fornecem uma interface de programação de aplicação, chamada API, possibilitando que aplicações externas possam interagir com a infraestrutura criada dentro do orquestrador. Dessa forma, é possível para um time de desenvolvimento criar uma aplicação que gerencia sua infraestrutura virtual da maneira que lhe for desejada.

Atualmente, o IFSC campus São José está migrando todos os seus serviços oferecidos para uma nuvem privada de contêineres. Tais serviços vão desde fóruns como *Moodle* e *Wiki*, acessada por alunos e professores e utilizada para troca de material acadêmico assim como informações gerais de disciplinas, até aplicações como o *Matlab* ou *Octave*. Segundo a CTIC do IFSC campus São José, a implementação de tais serviços em uma rede de contêineres orquestrada por Kubernetes trouxe melhorias significativas no que diz respeito a rotina de implantação de novos serviços e da manutenção de serviços já existentes que foram migrados migrados para a nova arquitetura. Porém, a constante manutenção de tais serviços, assim como a divisão de trabalhos e turnos dos profissionais do setor, faz com que mesmo as ferramentas de um orquestrador não sejam o suficiente para deixar a rotina de trabalho simples e prática.

Uma das necessidades da CTIC do IFSC campus São José é ter a possibilidade de utilizar uma ferramenta já utilizada pela equipe para poder gerenciar com mais praticidade o orquestrador de serviços, por questões de conveniência. Atualmente, a equipe utiliza, para comunicação interna, uma aplicação de mensagens instantâneas de texto. Tais aplicações de chat são muito comuns atualmente em empresas de tecnologia devido a atual preferência pelo uso de texto na comunicação (BURKE, 2016) associada a possibilidade do uso de extensões que permitam a execução de tarefas que não são de responsabilidades da ferramenta de chat em si. Uma extensão que vem se tornando comum atualmente nesse tipo de aplicação é o de *ChatBot*.

Um *ChatBot* é uma aplicação que pode automatizar uma conversação com um ser humano. Seja por uso de linguagem natural ou por comandos, é um programa que é capaz de entender o que a pessoa que interage com ela quer, e tomar ações baseadas nessa interação (HOGLE, 2015). Apesar de não ser um conceito recente, *ChatBot*s vêm sendo amplamente implementados por diversos serviços na área de desenvolvimento porque permitem facilidades na integração com sistemas de controle de projetos e de versionamento de códigos de programação (HAND, 2016). No ambiente de bate-papo do trabalho, podem-se demandar tarefas para o *ChatBot* e ele executará o que se deseja. Com a popularização dos *ChatBot*s, surgiram diversas bibliotecas que implementam não só o programa responsável por interagir com ferramentas de conversação de texto, mas que também oferecem APIs em diversas linguagens para que se personalize o próprio *ChatBot*.

Existe, atualmente, uma demanda por parte da CTIC para a implementação de uma aplicação que seja capaz de monitorar os serviços executados em contêineres, notificar

a equipe de potenciais problemas no funcionamento de tais serviços e permitir a execução de tarefas simples como a finalização, a reinicialização e a duplicação dos mesmos. Tendo em vista tal demanda, esse trabalho pretende entregar uma solução onde a equipe possa, através do seu aplicativo de mensagens via texto, monitorar e receber notificações do estado de todos os seus serviços hospedados no orquestrador, assim como executar as tarefas simples de exclusão, replicação e reinicialização de tais serviços. Tal solução será dada em forma de um *software*, e mediante necessárias adaptações, será extensível a qualquer aplicação de mensagem de texto que seja capaz de receber a funcionalidade de *ChatBot*.

1.1 Objetivos

1.1.1 Objetivos Gerais

O objetivo deste trabalho é desenvolver um *ChatBot* integrável a ferramentas de chat online, capaz de entregar ao usuário monitoramento de estado de serviços, notificações de alertas e erros em serviços e acesso aos comandos de remoção, duplicação e reinicialização de serviços utilizando a API disponibilizada pelo orquestrador.

1.1.2 Objetivos Específicos

- Criação de uma linguagem de comandos para o ChatBot.
- Criação de novas ações do *ChatBot*, programadas pelo usuário a partir da aplicação de chat (macros).

2 Fundamentação Teórica

2.1 Virtualização

A internet tem sido um sucesso sem precedentes nas últimas três décadas, e vem crescendo de maneira impressionante. Porém, sua popularidade se tornou um dos principais problemas relacionados ao seu próprio crescimento, já que devido a sua natureza multiprovedora, a adoção de novas arquiteturas ou modificações nas arquiteturas existentes requer o consenso de todos os envolvidos no seu desenvolvimento e impactaria bilhões de usuários ao redor do mundo (CHOWDHURY; BOUTABA, 2010). Além disso, a grande demanda de recursos impostas por diversas aplicações na internet força a infraestrutura a se tornar cada vez mais eficiente na operação de tais aplicações. Virtualização é uma boa alternativa para esses problemas, pois fornece maior flexibilidade, menor custo, escalabilidade e melhor utilização de recursos, além de eficiência energética (BARI et al., 2013).

O conceito de máquina virtual surgiu na década de 1960, época em que a IBM desenvolvia o sistema operacional experimental `M44/44X` - a partir dele, a IBM desenvolveu vários sistemas comerciais suportando virtualização, entre os quais o famoso `OS/370` (MAZIERO, 2014). Mais recentemente, a constatação de que o uso de máquinas reais gera um grande desperdício de recursos fez com que as atenções se voltassem para técnicas de virtualização. Um computador virtual é uma representação lógica de um computador real, em *software*. Ao desacoplar o *hardware* do sistema operacional, a virtualização provê maior flexibilidade operacional e aumenta a taxa de utilização da camada física do computador (EDUCATION, 2007). *Virtual Machine* (VM) pode se descrita como uma abstração de *hardware* físico, transformando um servidor em vários outros. O *Hypervisor* ou *Virtual Machine Monitor* (VMM) é a aplicação responsável por criar, executar e permitir que múltiplas VMs coexistam numa mesma máquina real (DOCKER, 2018). Uma ilustração de uma arquitetura de virtualização é exibida na Figura 2.

Tecnologias de virtualização se tornaram muito populares nos anos mais recentes, trazendo diversas soluções de *software* (como Xen, VMware e KVM) e outras incorporações de suporte de *hardware* (como o Intel-VT e o AMD-V). Porém, mesmo com todos os benefícios que oferece, o uso de virtualização vem sendo evitado em alguns cenários, principalmente na maioria das instalações de *High Performance Computing* (HPC) devido a sua grande sobrecarga de desempenho (*overhead*). Porém, implementações recentes de virtualizações baseadas em contêineres (como o *Linux Conteiner* (LXC)) oferecem uma camada de virtualização leve, com desempenho se equiparando a um computador nativo (XAVIER et al., 2013).

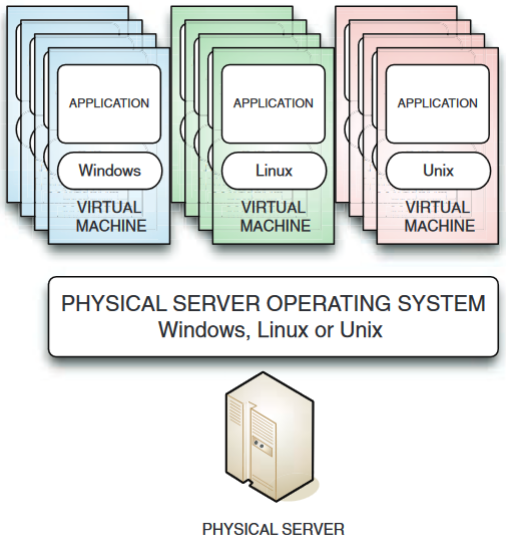

Figura 2 – Modelo de virtualização de servidores (EDUCATION, 2007).

2.2 Contêiner

Seguindo um caminho diferente ao das máquinas virtuais gerenciadas por *hypervisors*, a virtualização baseada em contêiner oferece uma forma diferente de abstração. É considerada uma alternativa mais eficiente, implementando um isolamento de processos ao nível do sistema operacional, evitando assim a sobrecarga (*overhead*) que acompanha a implementação de VMs. Os contêineres são executados, ao mesmo tempo, sobre o mesmo sistema operacional do servidor físico, e um ou mais contêineres podem ser executados sobre o mesmo sistema operacional base (JOY, 2015).

Atualmente, a implementação mais popular de contêineres é o *Docker*, uma plataforma desenvolvida por empresa de mesmo nome, que entrega independência entre aplicações, infraestrutura e desenvolvedores (DOCKER, 2018).

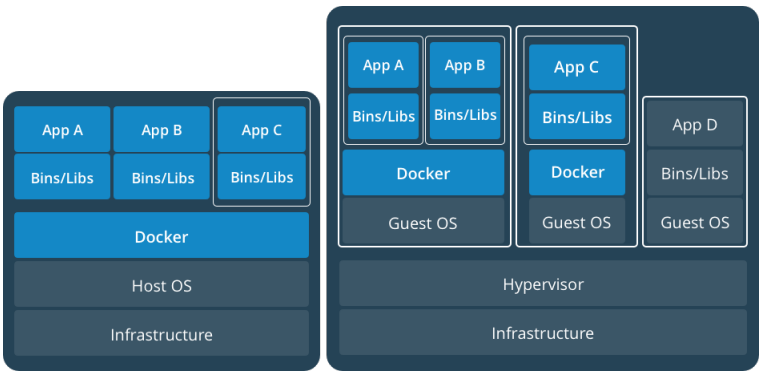

Figura 3 – Ilustração comparando a arquitetura da Máquina Virtual e de Contêineres *Docker* (DOCKER, 2018).

2.3 *Kubernetes*

Kubernetes é um orquestrador de contêineres com código aberto desenvolvido para simplificar a tarefa de construir, implantar e manter sistemas distribuídos escaláveis na nuvem (HIGHTOWER; BURNS; BEDA, 2017). De acordo com a documentação oficial (The Kubernetes Authors, 2018): "*Kubernetes* é uma plataforma de código aberto portável e extensível para gerenciamento de serviços, que facilita tanto a configuração declarativa quanto a automação. Possui um ecossistema grande e de rápido crescimento, e seus serviços, suporte e ferramentas estão amplamente disponíveis".

Todas as informações a seguir referentes ao *Kubernetes* foram extraídas da documentação (The Kubernetes Authors, 2018).

Para trabalhar com o *Kubernetes*, definem-se os objetos da API do *Kubernetes* para descrever o estado desejado do seu *cluster*: quais aplicações você deseja executar, qual a imagem de contêiner utilizar, o número de réplicas, que rede e recurso de disco se deseja disponibilizar, e mais opções. Isso é feito tipicamente através comandos via *Command-line Interface* (CLI)[1]. Também é possível utilizar diretamente e API para interagir com o *cluster* e definir ou modificar o estado desejado.

Com o estado desejado configurado, o **Painel de Controle do *Kubernetes*** trabalha para garantir que o estado do *cluster* em execução permaneça equivalente ao estado descrito em configuração. Para isso, são feitas diversas ações automaticamente, entre elas iniciar e reiniciar contêineres e escalar o número de réplicas de uma determinada aplicação e versão. O **Painel de Controle do *Kubernetes*** consiste em uma coleção de processos rodando no *cluster*:

- *The Kubernetes Master*: coleção de três processos que rodam em um ou mais nós do *cluster*. Os processos são: `kube-apiserver`, `kube-controller-manager` e `kube-scheduler`.
- Nó *non-master*: possuem os processos `kubelet`, que se comunica com o nó *master* e `kube-proxy`, que se reflete em um serviço de rede em cada nó.

2.3.1 Objetos do *Kubernetes*

Kubernetes contém um número de abstrações que representam o estado de um sistema: aplicações em contêineres implantadas, suas redes associadas e recursos de discos, e outras informações sobre o que o *cluster* está fazendo. Essas abstrações são representadas por objetos da API do *Kubernetes*. Abaixo, segue explicação dos principais objetos segundo documentação oficial do *Kubernetes*.

[1] O aplicativo `kubectl`.

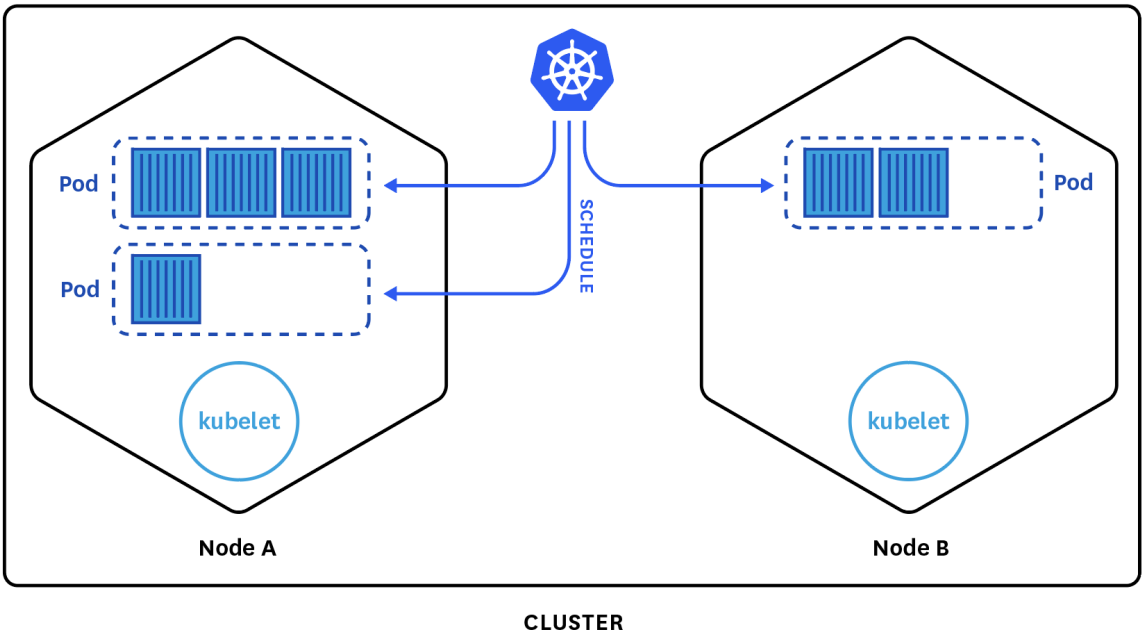

Figura 4 – Componentes de um cluster no *Kubernetes* (The Kubernetes Authors, 2018). Dentro de um nó do Kubernetes podemos observar os *Pods* e o serviço *kubelet*.

2.3.1.1 *Pods*

Um *Pod* é a menor e mais simples unidade no modelo de objetos do *Kubernetes* que se é possível criar ou implantar. Ele representa um processo rodando no seu *cluster*. Um *Pod* encapsula uma aplicação em contêiner (ou, em alguns casos, múltiplos contêineres), recursos de armazenamento, um IP de rede único e opções que especificam como controlar o contêiner. Ele representa uma unidade de implantação: uma única instância de uma aplicação no *Kubernetes*, que pode ser um contêiner ou um pequeno número de contêineres que estão acoplados e compartilham dos mesmos recursos.

Existem três tipos do que se chama Controlador de *Pod*:

- *Deployment*: Descreve-se o estado desejado em um *objeto de Deployment*, e o controlador de implantação (*Deployment Controller*) muda o estado atual do *Deployment* para o estado desejado. Você pode definir *Deployments* para criar novos conjuntos de réplica, ou remover *Deployments* existentes e adotar todos os seus recursos com um novo *Deployment*.

- *StatefulSet*: Se comporta como o *Deployment*, mas mantém a identidade de cada *Pod* que possui. Tais *Pods* são criados com as mesmas especificações, mas não são intercambiáveis: cada um tem um identificador de persistência, que mantém através de qualquer re-escalonamento.

- *DaemonSet*: garante que todos, ou alguns nós executem uma cópia de um *Pod*. Se um nó foi adicionado ao *cluster*, *Pods* são adicionados a ele. Quando os nós são

destruídos, os *Pods* são removidos por um coletor de lixo (*garbage collector*).

Docker é a implementação de contêiner mais utilizada em um *Pod*, porém outras implementações também são suportadas pelo *Kubernetes*.

2.3.1.2 Serviços

Pods são criados e destruídos a todo momento, e quando eles morrem nunca mais são ressurretos. Sua criação e destruição é feita dinamicamente pelas *ReplicaSets*. Isso gera um problema: se um *Pod* específico provê recursos ou funcionalidades para outros *Pods*, como esses outros *Pods* sempre sabem onde buscar seus recursos, uma vez que valores como endereço IP podem, assim, mudar?

Esse problema é resolvido por serviços do *Kubernetes*, que são abstrações que definem um conjunto lógico de *Pods* e uma política de como acessar eles. O conjunto que é alvo de um serviço é usualmente determinado por um *Seletor de Label*. Logo, cada *Pod* tem uma ou mais etiquetas de identificação, ou *label*.

2.3.1.3 *Namespaces*

Namespaces são *clusters* virtuais suportados pelo mesmo *cluster* físico. É uma maneira de dividir recursos de um *cluster* entre múltiplos usuários. A intenção é de que *namespaces* sejam utilizados em ambientes com muitos usuários espalhados por diversos times de desenvolvimento ou projetos.

2.3.2 Painel de Controle do *Kubernetes*

As várias partes do Painel de Controle do *Kubernetes*, como o *Kubernetes Master* e o processo *kubelet*, governam como o *Kubernetes* se comunica com o *cluster*. O painel de controle mantem gravado todos os **objetos *kubernetes*** do sistema, e executa contínuas rotinas para gerenciar o estado desses objetos. A qualquer dado tempo, o Painel de Controle vai responder a alterações feitas no *cluster* e trabalhar para fazer com que o estado atual de todos os objetos do sistema se igualem o novo estado desejado.

2.3.3 API

API é um conjunto de métodos claramente definidos de comunicação entre o serviço e qualquer outro *software* ou componente. Uma API define a estrutura de dados de um programa (BLOCH, 2006).

No centro do Kubernetes existe uma API, e tudo na plataforma é acessível através dessa API[2]. Tarefas como a criação e destruição de *Pods*, serviços e conjunto de réplicas

[2] Usando para tal *endpoints* específicos.

são todos traduzido em chamadas apropriadas de uma API *Representational state transfer* (REST) (MSV, 2016), um tipo de API que usa requisições *Hypertext Transfer Protocol* (HTTP) para gerenciar informações (MASSE, 2011).

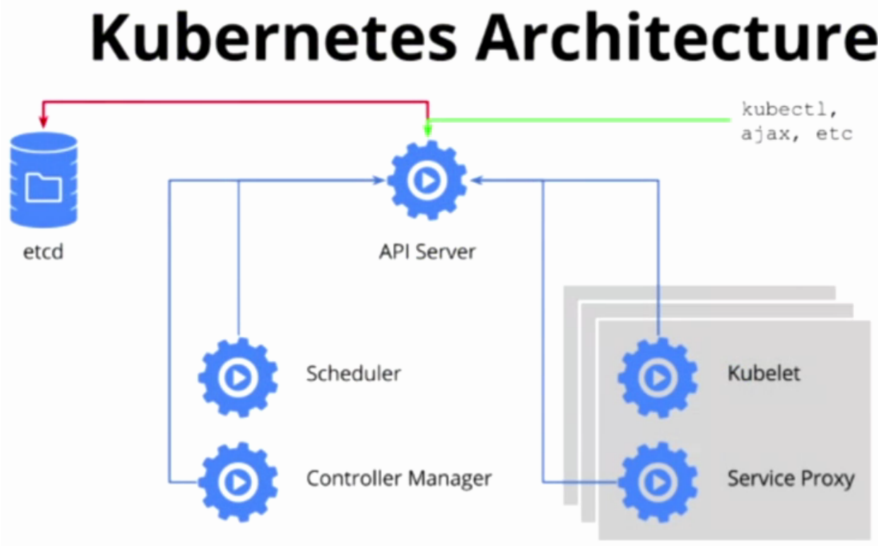

Figura 5 – Arquitetura da API do *Kubernetes* (The Kubernetes Authors, 2018).

2.4 Chatbot

De acordo Burke (2016), 913 milhões de textos são enviados por hora ao redor do mundo, e as aplicações Facebook e WhatsApp juntas processam 60 bilhões de mensagens por dia segundo Goode (2016). Segundo Shapira (2010), os jovens consideram mensagens de textos menos intrusivas e lhes dão mais controle numa conversação, já que não implicam o imediatismo da resposta de uma ligação de áudio ou vídeo. O uso de mensagens por texto torna a informação mais fácil de ser consultada diversas vezes e proporciona uma comunicação mais clara. Além do mais, processar um texto e fazer a auditoria de tal informação é muito mais simples do que executar o mesmo processo em mídias como áudio e vídeo. De acordo com Shapira (2010), essa é uma das principais causas para a popularização dos chats na atualidade.

2.4.1 Chat

Ferramentas de *chat*, ou *Instant Messaging* (IM), são um conjunto de tecnologias de comunicação usadas para realizar uma conversação baseada em texto entre dois ou mais participantes através da Internet, ou outros tipos de rede.

Segundo (HAND, 2016), "ferramentas populares de chat em grupo permitiram um aumento na transparência sobre o que acontece no decorrer do dia de uma equipe ou organização. Ao entrar em conversas em um espaço compartilhado onde outros podem

participar da discussão, uma maior eficiência e consciência é provida a uma maior parte da equipe.“

2.4.2 ChatBot

Um *ChatBot* é uma aplicação que pode automatizar uma conversação com um ser humano (HOGLE, 2015). O termo foi originalmente cunhado por Michael Mauldin em 1994 para descrever esses programas de conversação (MAULDIN, 1994). Hoje em dia, a maioria dos *Chatbots* são acessados por assistentes virtuais, como o *Google Assistant* ou o *Amazon Alexa*, via aplicativos de chat como o *Facebook Messenger* ou o *WeChat*, ou via aplicativos individuais e *websites* de organizações (ORF, 2016). Os *ChatBots* podem ser classificados em categorias de uso, como comércio conversacional (comércio eletrônico via bate-papo), análise, comunicação, suporte ao cliente, design, ferramentas para desenvolvedores, educação, entretenimento, finanças, alimentos, jogos, saúde, gestão de pessoas, marketing, notícias , produtividade, compras, social, esportes, viagens e utilitários (BARON, 2017).

2.4.3 Linguagem de comandos

Um *ChatBot* implementado sob linguagem de comandos é um *software* que responderá apenas a interações específicas por parte do usuário. São implementações mais simples, e garantem uma segurança maior, já que a necessidade de uma entrada por comandos específicos exclui eventuais tomadas de ações erradas (HAND, 2016).

```
1 $ @bot deletar conta
2 $ Erro: Comando desconhecido.
3 $ @bot deletar conta mathiashls
4 $ Conta mathiashls deletada com sucesso.
```

Listagem 2.1 – Exemplo de interação com chatbot por linguagem de comandos.

2.4.4 Linguagem Natural

De acordo com (BATES, 1995), o processamento por *Natural-language processing* (NLP) é um ramo da ciência da computação e inteligência artificial que se preocupa em desenvolver interações entre computadores e seres humanos utilizando a linguagem natural (humana). Os principais desafios da NLP são reconhecimento de discursos e o entendimento e geração de linguagem natural. É principalmente utilizada quando se deseja passar para o usuário a experiência de se estar conversando com uma entidade inteligente, e não um simples programa de computador.

```
1 $ @bot deletar conta
2 $ Qual o nome de usuario da conta que voce deseja deletar?
```

```
$ @bot mathiashls
$ Deletando conta de usuario mathiashls...
$ Conta deletado com sucesso.
```

Listagem 2.2 – Exemplo de interação com chatbot por linguagem natural.

2.4.5 ChatOps

ChatOps é um movimento que buscar usar ferramentar de *chat* em grupo para ir além da conversação básica, justapondo a discussão ao contexto e às ações tomadas dentro da própria ferramenta de chat. Ao criar uma interface unificada para times tomarem ações, acessarem informações relevantes e discutirem tudo isso uns com os outros, *ChatOps* entrega muitos benefícios para equipes e organizações.

Muitos dos princípios desse movimento derivam do *DevOps*, termo que deriva da junção das palavras *development* e *operation*, e é também um movimento recente, que busca criar uma ponte entre os segmentos operacionais e de desenvolvimento, melhorando a comunicação, a colaboração e o trabalho em equipe de uma equipe ou corporação. Também consiste em um método de desenvolvimento com ênfase na entrega de *software*, implantação automatizada, integração contínua e garantia de qualidade (JABBARI et al., 2016).

2.4.5.1 Integração com Aplicações

Muitos dos serviços que são usados diariamente tornam possível a interação e até troca de dados (leitura e escrita) com aplicações através das já discutidas anteriormente APIs. Dessa maneira, programas podem receber e enviar todo tipo de dado de várias aplicações de interesse de um time, tornando a ferramenta de chat ainda mais útil no ambiente de trabalho (HAND, 2016).

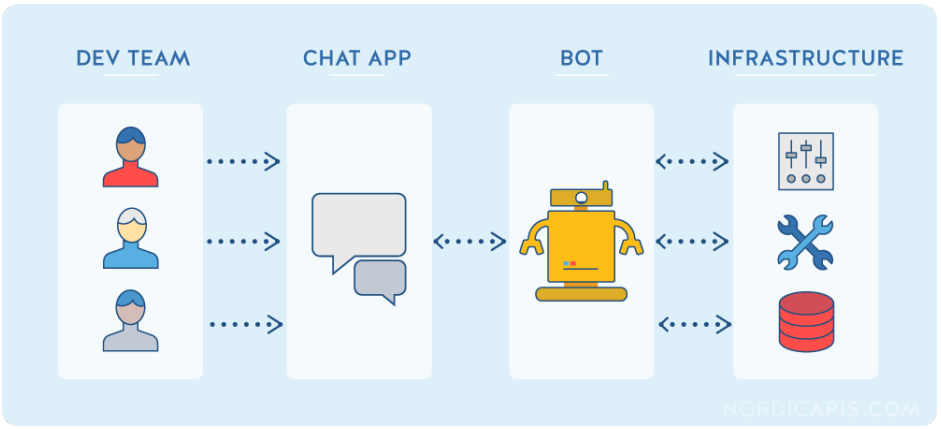

Figura 6 – Integração de ferramentas de chats com serviços diversos, através de *chatbots* (ATALAY, 2017).

3 Proposta

Segundo a CTIC do IFSC campus São José, grande parte dos serviços de nuvem oferecidos aos alunos são executados como contêineres Docker, gerenciados pelo Kubernetes, e os que ainda não se encontram nesse formato estão sendo migrados para tal. Essa alteração trouxe um maior aproveitamento dos recursos disponíveis para a equipe. Porém, segundo os mesmos, a escala atual de serviços faz com que o gerenciamento diário seja algo extremamente burocrático. Atualmente, os serviços disponibilizados são (CTIC-SJE, 2018):

- Ansible AWX
- MatLab
- MediaWiki
- Mosquitto
- Netbox
- NyqLab
- Octave
- OnlyOffice
- OpenLDAP
- OpenProject
- RocketChat
- ShareLatex
- Wordpress
- Zabbix

A equipe da CTIC do IFSC campus São José, além de utilizar Kubernetes para gerenciar todos os seus contêineres, utiliza ferramentas de chat online que permitem a integração de módulos, como por exemplo um Chatbot.

Tendo isso em mente, a presente proposta tem como objetivo implementar uma solução em formato de *software Chatbot* por linguagem de comandos, que permita ser

integrado a diversas ferramentas de chat online, e garanta ao seu usuário a capacidade de monitoramento de estado de serviços, notificações de alertas e erros em serviços, e acesso aos comandos de remoção, duplicação e reinicialização de serviços, utilizando a API disponibilizada pelo orquestrador.

Como o cliente em questão e maior interessado é a equipe da CTIC do IFSC campus São José, a implementação neste trabalho será realizada em torno da infraestrutura oferecida pelos mesmos. Sendo assim, será utilizado como orquestrador de contêineres o *Kubernetes*, e como ferramentas de chat o *Telegram* e o *Slack*.

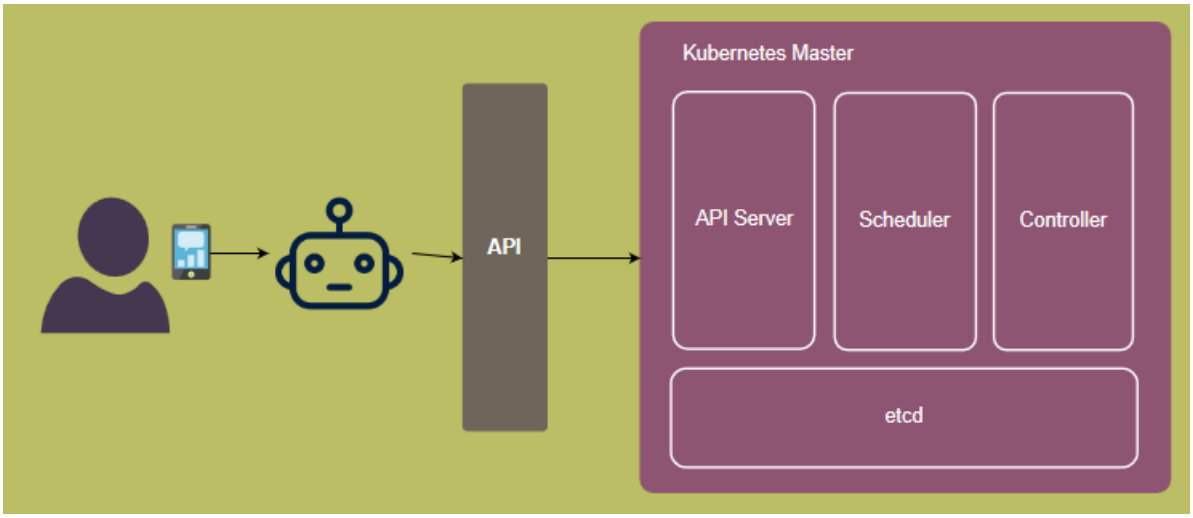

Figura 7 – Visão geral da topologia da proposta
O usuário não mais interage diretamente com a API do Kubernetes através de comandos compostos, mas sim com o *Chatbot*, na comodidade do seu chat utilizando comandos simples.

Para a implementação do software Chatbot será utilizada a linguagem de programação Python 3.6.6rc1 (ROSSUM; DRAKE et al., 2009) em conjunto com a biblioteca `Errbot` (ERRBOT, 2018), uma ferramenta de auxílio na criação de *Chatbots*, permitindo o programador concentre no desenvolvimento da lógica de negócio específica do seu bot, contando com um conjunto de *backends* para diversas ferramentas de chat.

O *Chatbot* será desenvolvido seguindo a sintaxe de comandos, já que o uso de linguagem natural adicionaria um nível de complexidade ao trabalho sem justificar qualquer ganho. Pelo contrário, usar linguagem natural acrescentaria pontos de falha no sistema, já que um comando mal interpretado poderia causar, por exemplo, a destruição de um serviço de forma equivocada. Por conta disso, o bot obedecerá a um conjunto de comandos específico, sem margem para interpretação por parte do programa.

Dentro do programa, existirá um mapeamento de palavras chaves, e uma lógica de quebra de comandos e identificação dos mesmos. Um comando não identificado da maneira correta produzirá uma mensagem de erro para o usuário. Um comando reconhecido com sucesso será traduzido para um comando da API do orquestrador.

```
# O comando enviado para o bot:
$ @ctic_bot del matlab

# Sera traduzido para um comando que a API do Kubernetes entenda:
$ kubectl delete pods matlab
```

Listagem 3.1 – Tradução de comando do chatbot para comando do Kubernetes

Será implementada, também, a opção de programar o *Chatbot* através do próprio chat. Para tal, será adicionada ao programa a opção de modo de desenvolvedor, onde se poderá entrar com o comando simplificado (que será passado para o *Chatbot* pelo usuário) e o que tal comando significa para o orquestrador (comando composto do Kubernetes, por exemplo). O *Chatbot* será capaz de analisar as entradas e, após verificar que os comandos são válidos e não nocivos (de acordo com as regras de negócio decididas no desenvolvimento do produto), o comando será adicionado ao bot. Caso contrário, o comando será descartado.

Uma característica interessante do sistema é que o software final será mantido como um contêiner, gerenciado pelo mesmo orquestrador que irá interagir. Esse modelo é perfeitamente possível e seguro, devido a maneira com que o orquestrador lida com os seus serviços. Imaginando um cenário onde o usuário peça para reiniciar o serviço do *Chatbot* através do próprio *Chatbot*, o mesmo irá parar de responder pelo tempo de reinicialização, e em seguida retornará a interagir com o usuário. No caso de uma remoção do serviço, ele irá parar de interagir com o usuário, mas entende-se que esse é o comportamento esperado. Existirá também a possibilidade de bloquear o acesso do usuário a serviços específicos, podendo ser o serviço do *Chatbot* um deles.

3.1 Plano de Trabalho

Aqui estão descritos os planos de trabalhos já realizados durante o período de elaboração da proposta e também os planos futuros de desenvolvimento.

1. Encontros com a equipe da CTIC do IFSC campus São José para completo entendimento da problemática do cliente.
2. Estudo de métodos de virtualização com ênfase em contêineres Docker.
3. Estudo aprofundado da ferramenta de orquestração utilizada pelo cliente (*Kubernetes*).
4. Instalação de ambiente de desenvolvimento com Kubernetes para desenvolver perícia com a ferramenta.
5. Desenvolvimento de conhecimento básico sobre escrita de *Chatbot*.

6. Avaliar como é dada a comunicação entre o módulo do *Chatbot* e as ferramentas de chat.

7. Validação da ideia (prova de conceito) com base nos três itens anteriores.

8. Escrita da proposta de Trabalho de Conclusão de Curso, contendo resumo, introdução, fundamentação teórica e proposta de trabalho, e entrega deste para a banca. Preparação da apresentação, defesa e correção do documento.

9. Iniciar a implementação do *Chatbot*, integrando o mesmo às ferramentas de chat e à API do Kubernetes.

10. Implementar autenticação de usuários, fazendo uso do *Lightweight Directory Access Protocol* (LDAP).

11. Desenvolvimento de lógica para que o *Chatbot* seja programado a partir do chat, pelos usuários.

12. Implantação na estrutura do IFSC câmpus São José para adaptação ao cenário real e testes por partes da CTIC.

13. Escrita do documento do TCC (Introdução, Proposta de trabalho, Fundamentação teórica, Implementação, Análises, Resultados e Conclusões)

	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
1	×	×	×								
2			×	×							
3			×	×							
4				×							
5				×							
6				×							
7				×	×						
8					×						
9						×	×	×			
10							×	×			
11								×	×		
12									×	×	
13										×	×

Referências

ATALAY, A. *ChatOps with Mattermost and Hubot*. 2017. Disponível em: <https://medium.com/@ahmetatalay/chatops-with-mattermost-and-hubot-59d0b141b220>. Citado 2 vezes nas páginas 15 e 34.

BARI, M. F. et al. Data center network virtualization: A survey. *IEEE Communications Surveys & Tutorials*, IEEE, v. 15, n. 2, p. 909–928, 2013. Citado na página 27.

BARON, J. *2017 Messenger Bot Landscape, a Public Spreadsheet Gathering 1000+ Messenger Bots*. 2017. Disponível em: <https://recast.ai/blog/2017-messenger-bot-landscape/>. Citado na página 33.

BATES, M. Models of natural language understanding. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 92, n. 22, p. 9977–9982, 1995. Citado na página 33.

BLOCH, J. How to design a good api and why it matters. In: ACM. *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*. [S.l.], 2006. p. 506–507. Citado na página 31.

BURKE, K. *73 Texting Statistics That Answer All Your Questions*. 2016. Disponível em: <https://www.textrequest.com/blog/texting-statistics-answer-questions/>. Citado 2 vezes nas páginas 25 e 32.

CHOWDHURY, N. M. K.; BOUTABA, R. A survey of network virtualization. *Computer Networks*, Elsevier, v. 54, n. 5, p. 862–876, 2010. Citado na página 27.

CTIC-SJE. *Serviços do Kubernetes no Repositório oficial da CTIC-SJE-IFSC no GitHub*. 2018. Disponível em: <https://github.com/ctic-sje-ifsc/servicos_kubernetes>. Citado na página 35.

DOCKER. *Comparing Containers and Virtual Machines*. 2018. Disponível em: <https://www.docker.com/what-container#/package_software>. Citado 3 vezes nas páginas 15, 27 e 28.

EDUCATION, I. Virtualization in education. *IBM Corporation, Whitepaper*, 2007. Citado 3 vezes nas páginas 15, 27 e 28.

ERRBOT. *Errbot Documentation*. 2018. Disponível em: <http://errbot.io/en/latest/index.html>. Citado na página 36.

GARFINKEL, T.; ROSENBLUM, M. When virtual is harder than real: Security challenges in virtual machine based computing environments. In: *HotOS*. [S.l.: s.n.], 2005. Citado na página 24.

GOODE, L. *Messenger and WhatsApp process 60 billion messages a day, three times more than SMS*. 2016. Disponível em: <https://www.theverge.com/2016/4/12/11415198/facebook-messenger-whatsapp-number-messages-vs-sms-f8-2016>. Citado na página 32.

GROUP, M. M. *Internet Growth Statistics 1995 to 2018*. 2018. Disponível em: <https://www.internetworldstats.com/emarketing.htm>. Citado na página 23.

HAN, B. et al. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, IEEE, v. 53, n. 2, p. 90–97, 2015. Citado na página 23.

HAND, J. *ChatOps – Managing Operations in Group Chat*. [S.l.]: "O'Reilly Media, Inc.", 2016. Citado 4 vezes nas páginas 25, 32, 33 e 34.

HIGHTOWER, K.; BURNS, B.; BEDA, J. Kubernetes: Up and running dive into the future of infrastructure. O'Reilly Media, Inc., 2017. Citado na página 29.

HOGLE, P. What is a chatbot? *learning Solutions*, 2015. Citado 2 vezes nas páginas 25 e 33.

JABBARI, R. et al. What is devops?: A systematic mapping study on definitions and practices. In: ACM. *Proceedings of the Scientific Workshop Proceedings of XP2016*. [S.l.], 2016. p. 12. Citado na página 34.

JOY, A. M. Performance comparison between linux containers and virtual machines. In: IEEE. *Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in*. [S.l.], 2015. p. 342–346. Citado na página 28.

LEINER, B. M. et al. A brief history of the internet. *ACM SIGCOMM Computer Communication Review*, ACM, v. 39, n. 5, p. 22–31, 2009. Citado na página 23.

MASSE, M. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. [S.l.]: "O'Reilly Media, Inc.", 2011. Citado na página 32.

MAULDIN, M. L. Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. In: *AAAI*. [S.l.: s.n.], 1994. v. 94, p. 16–21. Citado na página 33.

MAZIERO, C. A. Sistemas operacionais: Conceitos e mecanismos. *Livro aberto. Acessível em: http://wiki. inf. ufpr. br/maziero/lib/exe/fetch. php*, 2014. Citado na página 27.

MSV, J. *Take the Kubernetes API for a Spin*. 2016. Disponível em: <https://thenewstack.io/taking-kubernetes-api-spin/>. Citado na página 32.

NEWMAN, S. *Building microservices: designing fine-grained systems*. [S.l.]: "O'Reilly Media, Inc.", 2015. Citado na página 24.

ORF, D. *Google Assistant Is a Mega AI Bot That Wants To Be Absoutely Everywhere*. 2016. Disponível em: <https://gizmodo.com/google-assistant-is-a-mega-chatbot-that-wants-to-be-abs-1777351140>. Citado na página 33.

ROSENBLUM, M. The reincarnation of virtual machines. *Queue*, ACM, v. 2, n. 5, p. 34, 2004. Citado na página 24.

ROSSUM, G. V.; DRAKE, F. L. et al. *Python 3: Reference manual*. [S.l.]: SohoBooks, 2009. Citado na página 36.

SALEH, A. A.; SIMMONS, J. M. Technology and architecture to enable the explosive growth of the internet. *IEEE Communications Magazine*, IEEE, v. 49, n. 1, 2011. Citado na página 23.

SHAPIRA, I. Texting generation doesn't share boomers' taste for talk. *Washington Pos*, 2010. Citado na página 32.

STATS, I. L. *Internet Live Stats*. 2018. Disponível em: <https://www.internetlivestats.com/internet-users/>. Citado 2 vezes nas páginas 15 e 23.

The Kubernetes Authors. *Kubernetes Documentation*. 2018. Disponível em: <https://kubernetes.io/docs/home/>. Citado 4 vezes nas páginas 15, 29, 30 e 32.

XAVIER, M. G. et al. Performance evaluation of container-based virtualization for high performance computing environments. In: IEEE. *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*. [S.l.], 2013. p. 233–240. Citado na página 27.