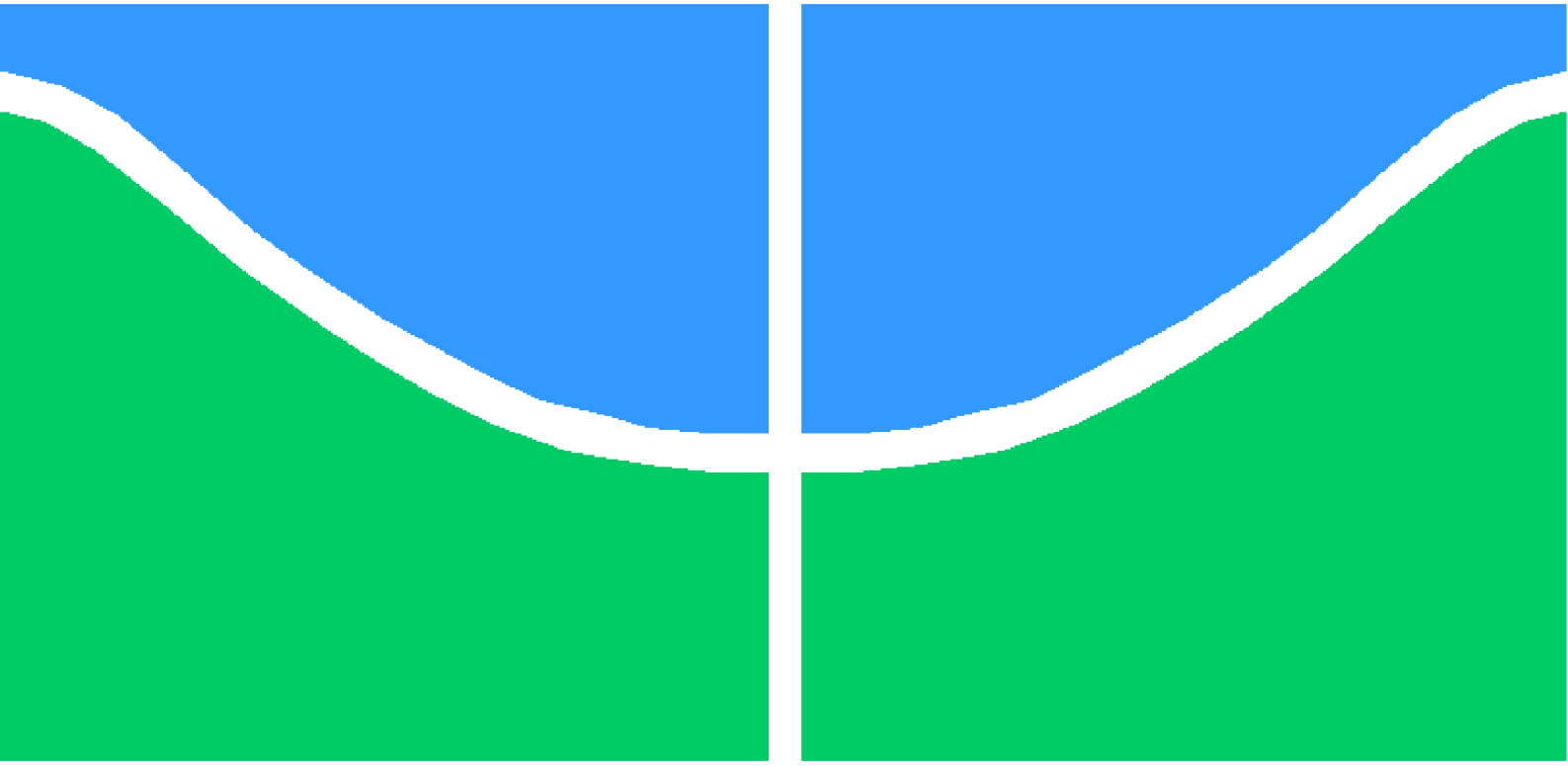

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Implantação automatizada de aplicações web em ambientes Debian GNU/Linux

Autor: Thiago de Souza Fonseca Ribeiro
Orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Brasília, DF
2016

Thiago de Souza Fonseca Ribeiro

Implantação automatizada de aplicações web em ambientes Debian GNU/Linux

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Coorientador: Dr. Antonio Soares de Azevedo Terceiro

Brasília, DF

2016

Thiago de Souza Fonseca Ribeiro
Implantação automatizada de aplicações web em ambientes Debian GNU/Linux/ Thiago de Souza Fonseca Ribeiro. – Brasília, DF, 2016-
63 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2016.

1. Implantação de Software. 2. Debian. I. Prof. Dr. Paulo Roberto Miranda Meirelles. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Implantação automatizada de aplicações web em ambientes Debian GNU/Linux

CDU 02:141:005.6

Thiago de Souza Fonseca Ribeiro

Implantação automatizada de aplicações web em ambientes Debian GNU/Linux

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Trabalho aprovado. Brasília, DF, 25 de julho de 2016:

Prof. Dr. Paulo Roberto Miranda Meirelles
Orientador

Dr. Fabio Macedo Mendes
Convidado 1

Msc. Renato Coral Sampaio
Convidado 2

Brasília, DF
2016

Agradecimentos

Agradeço primeiramente a Deus, pela minha vida e por alimentar a minha fé, por me dar forças para continuar e sabedoria para tomar as decisões certas.

Agradeço aos meus professores da Universidade de brasília, da Faculdade UnB Gama, por todo conhecimento comartilhado e pela importância de cada um na minha graduação, em especial, gostaria de fazer um agradecimento a alguns professores, pela sua importância, não só na minha formação profissional, como também na formação do meu caráter e suas contribuições na minha vida.

Agradeço ao meu orientador, Prof. Paulo Roberto Miranda Meirelles, por toda confiança que foi creditada em mim nesses últimos anos, pela sua paciência e dedicação na minha formação de engenharia de software e por sua dedicação ao curso de engenahria de software , por todos os ensinamentos, que são muito importantes para minha vida profissional e pessoal, obrigado pelo apoio e amizade, pelas portas que foram abertas e pela orientação neste trabalho.

Agradeço ao meu professor, Prof. Hilmer Rodrigues Neri, também pela confiança que foi creditada em mim, por ser o meu primeiro primeiro professor a acreditar no meu potencial, por todos os ensinamentos e sua dedicação ao curso de egenahria de software, agradeço pois muito do que sei tem a sua contribuição, tudo que conquistei até hoje foi graças a primeira porta que você abriu, agradeço pelos ensinamentos e por toda a dedicação e pelo apoio durante a minha graduação, obrigado pela amizade e por sempre confiar em mim aonde estiver.

Agradeço ao meu professor, Prof. Edson Alves da Costa Junior, por todos os ensinamentos, não só em disciplinas como na vida, tive a sorte de ter aula com um grande mestre, no qual nunca esquecerei cada ensinamento e cada momento de aprendizado. Obrigado pelo apoio e por ser sempre disponível a ajudar, por ser um excelente mestre, pela sua dedicação no curso de engenahria de software e por sua companhia nos cafés do obelisco.

Agradeço ao Antônio Terceiro, por todos os ensinamentos e toda confiança depositada em mim, pelas portas que foram abertas e por compartilhar seu conhecimento comigo, por toda paciência e dedicação que teve ao me ensinar muito do que sei, por ser um grande engenheiro de software, e um exemplo de profissional que quero seguir, que um dia eu pretendo ser ao menos um pouco do que você é.

Agradeço à todos os membros do Laboratório Avançado de Produção, Pequisa e Inovação em Software - LAPPIS pelas experiências compartilhadas, o crescimento con-

junto e o trabalho em equipe. Sei que dificilmente encontrarei uma equipe tão brilhante como essa, por isso agradeço pelos momentos nesses três anos de LAPPIS, foram momentos de muita alegria, desafios e aprendizado, espero um dia poder trabalhar com vocês novamente.

Agradeço a minha família, principalmente aos minha mãe Rosilene e meu pai Haroldo, por toda sua dedicação e amor, pela confiança e pelo apoio incondincional em todos os momentos da minha vida, por serem minha maior motivação e meu espelho como ser humano, por toda admiração e todo investimento. A minha conquista também é a conquista de vocês, e agradeço por serem excelentes pais e grandes profissionais em suas áreas de atuação, isso me faz querer ser o melhor, assim como vocês.

Agradeço também aos meus familiares e amigos que me apoiaram e me ajudaram a chegar até aqui, principalmente tios, amigos, primos, avós, e minha namorada Camila, por sempre acreditarem em mim e por sempre me motivarem a não desistir nunca.

"Deus não coloca um peso nos ombros de um homem se souber que ele não pode carregá-lo. Muhammad Ali

Resumo

A implantação de aplicações web de grande escala apresentam vários desafios, com isso, a implantação automatizada de software vem se tornando uma necessidade, principalmente pelos desafios de instalação e configuração das ferramentas web mais modernas, podendo facilitar a instalação e configuração de aplicações, tanto para desenvolvedores como para usuários. Este trabalho trata da implantação de software com seus procedimentos e ferramentas, com o foco na automação da instalação de aplicações web em ambientes Debian GNU/Linux, para isso foi feito uma colaboração da construção da ferramenta Shak, que propõe a instalação de aplicações web com apenas uma instrução, o que possibilita a instalaçaõ e configuração de ferramentas web com pacotes disponíveis nos servidores oficiais do Debian, utilizando protocolos seguros como HTTPS e implantação de aplicações web utilizando hospedagem virtual.

Palavras-chaves: Implantação de software, Debian.

Abstract

The deployment of large-scale web applications present several challenges, with it, an automated deployment of software is becoming a necessity, mainly for installation challenges and configuration of more modern web tools and facilitate the installation and configuration of applications, for both as developers and users. This work is about the software deployment with procedures and tools, with the focus on automation of web applications instalation on Debian GNU/Linux environments, for this, it made made the evolution of Shak tool, that offers a Web application installation with only instruction, which allows an installation and configuration tools web with available packages in official debian servers, using secure protocols, like HTTPS and Web application deployment using virtual hosts.

Key-words: Software deployment, Debian.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ACME	Automatic Certificate Management Environment
CSS	Cascading Style Sheets
DevOps	Amálgama de Desenvolvimento e Operações
DNS	Amálgama de Desenvolvedor e Operações
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IRC	Internet Relay Chat
OMG	Object Management Group
POP3	Post Office Protocol
SHAK	Self Hosting Applications Kit
SMTP	Simple Mail Transfer Protocol
SSL	Secure Socket Layer
SQL	Linguagem de consulta estruturada
TLS	Transport Layer Security

Sumário

1 Introdução

A internet emergiu no contexto da Guerra Fria na década de 60, em um projeto exército norte-americano. Os motivos de sua criação eram: criar um sistema de informação e comunicação em rede, que sobrevivesse a um ataque nuclear e dinamizar a troca de informações entre os centros de produção científica. Os militares pensaram que um único centro de computação centralizando toda informação era mais vulnerável a um ataque nuclear do que vários pontos conectados em rede, pois assim a informação estaria espalhada por inúmeros centros computacionais pelo país (GILES, 2010).

Desde a sua origem, a internet foi projetada para ser uma rede descentralizada e flexível (CASTELLS, 2003). A maioria dos protocolos utilizados na rede não dependem de pontos centrais para funcionar. De fato, é de conhecimento geral que, por exemplo, usuários com contas de e-mails em provedores diferentes podem trocar mensagens entre si sem qualquer problema. Considerando que no passado, a maioria dos softwares eram como uma ilha isolada hoje, os softwares são disponibilizados através da web ou conectado através da Internet, como por exemplo: as redes sociais e mercados de compras online (SPINELLIS, 2012a).

Essa grande sofisticação implica numa alta complexidade de desenvolvimento e manutenção, que faz com que a implantação de software e a manutenção de serviços exija conhecimento técnico especializado. Para organizações, isto implica num maior custo para manter sistemas e aplicações. Para indivíduos, isto quase sempre impossibilita a manutenção de serviços próprios e leva à procura por serviços gratuitos.

O mesmo vale para outras aplicações, em especial para aquelas aplicações que são utilizadas na web, como por exemplo: um blog, uma rede social ou site de notícias. Eles podem estar hospedados em qualquer provedor de internet, e mesmo os usuários de outros provedores, também terão acesso a elas.

Existe atualmente uma tendência de centralização da internet. Vaz mostra através de uma série de dados empíricos que a internet está centralizada (VAZ, 2006), A imensa maioria dos usuários individuais utilizam contas de e-mail em algum dos grandes provedores de e-mail, como gmail (TERCEIRO, 2015).

Isso também acontece com aplicações responsáveis por guardar dados dos usuários, com armazenamento de documentos na nuvem, documentos como: fotos, vídeos e arquivos, são geralmente armazenados em ferramentas conhecidas, como Google Drive ou Dropbox, estes serviços que não custam nada ao usuário, mas quando o usuário aceita os termos de uso, é possível que algumas empresas utilizem das informações pessoais dos usuários para usos comerciais.

Essa excessiva centralização traz problemas para os usuários que utilizam deste tipo de ferramentas, tais como: riscos a privacidade, subordinação dos usuários aos provedores e pontos centrais de falhas (TERCEIRO, 2015).

Para solucionar este problema seria necessário que existisse uma alternativa à centralização da internet, através de uma solução que elimina barreira técnica para que usuários interessados possam ter servidores próprios com aplicações, sem a necessidade de conhecimento técnico especializado.

A grande dificuldade de prover essa alternativa é que os sistemas voltado para a Internet consistem em muitas partes que são complicadas de gerenciar, alguns requisitos como disponibilidade e desempenho são indispensáveis, além de ser necessário gerenciar as aplicações em seus devidos servidores, gerenciamento de banco de dados, e configurações de recuperação de dados, além de administrar uma infinidade de bibliotecas de terceiros e serviços online (SPINELLIS, 2012b).

Além disso, também existe o impeditivo dos usuários terem que hospedar seus próprios serviços, porém, nos últimos anos este fator vem perdendo relevância em função de dois fenômenos. O primeiro deles é o barateamento do acesso a servidores virtuais privados decorrente os avanços da computação em nuvem. Por outro lado, a disponibilidade de servidores físicos de dimensão reduzida e baixo consumo de energia, em conjunto com o barateamento de conexões de banda larga à internet (TERCEIRO, 2015).

Esses impeditivos atingem de fato o usuário final, e a decisão acaba sendo de utilizar serviços gratuitos das grandes empresas, em troca de acesso às suas informações pessoais, cedidas de forma consciente ou não. Porém, de acordo com cada vez mais, caminhamos para uma mídia que atenda às necessidades do indivíduo. O usuário será o responsável por aquilo que deseja consumir na rede, essa mídia denomina-se You-Media *(U-Media)*. São inerentes colaboração à U-Media: contribuição e comunidades, participação e customização, além da descentralização dos serviços (TERRA, 2006).

Este trabalho relata a evolução da ferramenta Shak, que é uma ferramenta para implantação automatizada de aplicações web em sistema Debian GNU/Linux, que tem como objetivo fornecer uma alternativa à centralização da internet através de uma solução que elimina barreira técnica para que usuários interessados possam ter servidores próprios com aplicações sem a necessidade de conhecimento técnico especializado. O que problema que envolve este trabalho é:

Como implantar aplicações web em sistema Debian GNU/Linux de forma automatizada e segura?

1.1 Objetivo

O objetivo deste trabalho consiste na contribuição da construção de uma ferramenta que possa automatizar instalação e configuração de aplicações web em sistemas Debian GNU/LINUX, a partir de pacotes que sejam distribuídos oficialmente pelo Debian, mostrando os aspectos mais importantes que devem ser tratados durante todo o processo de configuração e instalação de um software, facilitando assim, que tanto usuários como desenvolvedores possam implantar aplicações com apenas uma instrução.

As contribuições deste trabalho são:

Contribuições Tecnológicas

1. **CT1** Implantação de sistemas Debian GNU/LINUX,
   a) Implantação de aplicações web com certificados digitais auto-assinados.
   b) Implantação de múltiplas aplicações web em um mesmo servidor utilizando hospedagem virtual.
   c) Implantação de automatizada de aplicações web utilizando pacotes distribuídos oficialmente pelo Debian.

Contribuições Científicas

1. **CC1** Gestão de configuração de software,
   a) Estudo teórico sobre implantação automatizada de software.
   b) Aplicação prática do modelo de implantação de software sugerido pela *Object Management Group (OMG)*.

1.2 Organização do trabalho

O trabalho está organizado na seguinte forma: O Capítulo 2 trás o referencial teórico necessário para apoio o desenvolvimento da solução, como a gerência de configuração de software, o processo de implantação de software e métodos e técnicas para implantação automatizada de software, já o Capítulo 3 fala sobre a definição e preparação dos estudos, que envolve os trabalhos relacionados, envolve também como será construído a solução e também como a solução será validada. Já no Capítulo 4 contém os resultados alcançados, e por fim as considerações finais no Capítulo 5.

2 Gestão de configuração de software

De acordo com IEEE (2012) gestão de configuração de software *GCS* é uma área da engenharia de software cujo objetivo é:

- Identificar e documentar as características funcionais de um produto.
- Controlar quaisquer alterações a estas características.
- Registrar e relatar cada mudança do seu estágio de implantação
- Apoiar a auditoria dos produtos, resultados, serviços ou componentes para verificar a conformidade com requisitos

Gestão de Configuração de Software é uma das capacidades fundamentais que devem estar em qualquer projeto de desenvolvimento de software. Ela é tradicionalmente dividida em quatro atividades, e são elas: configuração, identificação e controle, isto é encarado como uma ferramenta de gestão que podem ajudar na orientação de cada projeto, pois ajuda a manter a integridade do produto e manter o qualidade do projeto sob controle (BENDIX; KOJO; MAGNUSSON, 2011).

Ela também é essencial para a engenharia de software, pois estabelece e protege a integridade de um produto ao longo da sua vida útil, percorrendo através dos processos de desenvolvimento, teste e entrega do produto, bem como durante a sua instalação, operação, manutenção e eventual evolução(IEEE, 2012).

Gestão de configuração de software também fornece a infraestrutura que é a base para qualquer tipo de projeto de software. Isto facilita a coordenação e comunicação entre o vários participantes numa equipe de desenvolvimento de software. Dentro da gestão de configuração de software existem várias áreas de atuação, este trabalho aborda especificamente o segundo item dos objetivos citados acima, abordando a implantação de software(BENDIX; KOJO; MAGNUSSON, 2011).

2.1 Implantação de Software

Software só oferece valor para os clientes quando eles estão implantados em produção e assim possam proporcionar as funcionalidades necessárias ao usuário final, por isso a importância da fase de implantação de software, pois é nela em que a equipe de desenvolvimento disponibilizará o software ou uma atualização com novas funcionalidades ao usuário.

Sendo assim, a Implantação de software é um conjunto de atividade cruciais para todos os fornecedores de software, isto começa desde um pedido de um novo requisito de software até todas as medidas necessárias para essa nova versão fique disponível para o cliente (MANTYLA; VANHANEN, 2011). A implantação de software é composta por várias atividades que são essenciais para disponibilizar um produto a alguém, como por exemplo: instalação de dependências, arquivos de configuração e instalação da própria aplicação.

Uma grande aliada das equipes de desenvolvimento é a implantação automatizada de software, que se refere à prática de implantar o software para os usuários finais automaticamente, evitando qualquer tipo de execução de esforço manual, por isso, a prática de implantação automatizada facilita na rápida entrega de software, tanto para implantações de aplicações em servidores na nuvem, como implantação de software para usuários em seus computadores (RAHMAN et al., 2015).

As aplicações de software não são mais sistemas autônomos, são cada vez mais o resultado da integração de coleções de componentes, o que pode tornar a implantação de software um processo complicado, e nem sempre existir a garantia de que cada componente será implantado corretamente. Portanto, a equipe de desenvolvimento deve encontrar uma maneira de lidar com uma maior incerteza no ambiente nos quais seus sistemas vão operar, garantindo que a implantação do software seja feita da forma correta, evitando quaisquer acontecimentos de erros e imprevistos (CARZANIGA et al., 1998).

Todas essas questões devem ser levadas em consideração, e a atividade de implantação de software deve receber uma atenção especial, visto que os softwares estão ficando cada vez mais complexos, podendo ter várias ferramentas que são integradas, construídas em diferentes linguagens de programação e com diferentes configurações, com todas essas variáveis compondo uma implantação automatizada de software.

Este capítulo possui uma subseção sobre processo de Implantação de Software, para entender melhor como funciona o processo de implantação de software, depois uma subseção sobre *DevOps*, que busca trazer um conceito de implantação automatizada de software junto aos métodos ágeis, e por fim, uma seção sobre ferramentas e técnicas que buscam facilitar e resolver os problemas da implantação automatizada de software, com as características típicas que essas ferramentas trazem para tornar a implantação mais rápida e com qualidade.

2.1.1 Processo de Implantação de Software

A implantação de um software é o processo que vai desde a aquisição desse software até a sua execução (LEITE, 2014). O processo implantação consiste em diversas atividades que devem ser executadas desde o planejamento da implantação até a disponibilidade para

uso.

A *OMG* é uma organização internacional sem fins lucrativos que aprova padrões abertos para tecnologias e eles definem uma especificação de implantação e configuração de aplicações distribuídas baseadas em componentes (OMG, 2006). A *OMG* também diz que o processo de implantação é um processo que se inicia desde a aquisição de um componente até o momento em que esse componente está em plena execução, pronto pra uso.

Segundo OMG (2006) os principais termos definidos dentro de uma implantação de software são:

- **Implantador:** Pessoa ou equipe responsável pelo processo de implantação de um sistema.
- **Ambiente alvo:** São o servidor ou conjunto de servidores em que os componentes são implantados.
- **Nó:** É um recurso computacional onde se implanta um componente, por exemplo uma máquina virtual dentro do servidor que vai hospedar um serviço de banco de dados. Os nós fazem parte do ambiente alvo.
- **Pacote:** Artefato executável que contém o código binário do componente . É por meio de um pacote que um serviço pode ser instalado e executado dentro de um sistema operacional, e que são característicos dependendo da distribuição do sistema operacional, por exemplo: .deb para Debian e .rpm para Redhat, ou pacotes independentes de sistema operacional como por exemplo pacotes jar.

Além disso, é definido as fases que compõem o processo de implantação e de acordo com OMG (2006) as fases são:

- **Planejamento:** O planejamento da implantação é uma fase para identificar os componentes necessários na implantação e como cada um será distribuído entre os nós do ambiente alvo.
- **Preparação:** São os procedimentos necessários para preparar o ambiente alvo para que um determinado componente possa ser executado, isso envolve configuração do sistema operacional, instalação e configuração de dependências necessárias (por exemplo um servidor web como Apache ou Nginx) e a transferência do componente para o servidor onde ele será executado.
- **Instalação:** O implantador transfere o componente para a infraestrutura alvo, é quando instalamos um dado componente em um servidor, um exemplo disso é a instalação de uma aplicação via pacotes (.deb ou .rmp).

- **Configuração:** Edição de arquivos de configuração para alterar determinado comportamento de uma aplicação, o implantador aplica configurações específicas a aplicação a partir de arquivos de configuração.
- **Inicialização:** É quando a aplicação é iniciada e entra em execução, pronto para receber chamadas de seus clientes.

Os termos e fases formam uma estrutura básica que um processo de implantação de software deve conter, podendo haver modificações de acordo com a necessidade do implantador, cada fase é necessária para que possa ter uma implantação consistente, ou seja, as atividades dentro de uma implantação de software estarão organizadas, evitando assim possíveis erros durante a implantação, essa organização também podendo ajudar o time de implantação a encontrar os problemas ocorridos dentro de uma implantação de software, caso ocorram.

Dependendo do tamanho da aplicação essas fases podem se tornar tarefas complicadas, e com isso existe a necessidade de automatizar o processo de implantação, assim diminuindo a possibilidade de erros comparado a uma tarefa manual e consequentemente tornando o trabalho mais ágil, dando adeus aos longos manuais de instalação. O objetivo de um processo de implantação automatizado é proporcionar um processo de implantação reprodutível, confiável e fácil de ser executado (HUMBLE; FARLEY, 2010).

Para que a implantação automatizada de software seja possível, é necessário o uso de ferramentas para poder automatizar todo o processo de implantação, mas é necessário que o implantador compreenda cada fase do processo para que possa ser feito um bom planejamento e execução da implantação.

2.1.2 Desenvolvimento e Operações (DevOps)

Como resultado ao longo dos anos, o que aconteceu é que as equipes de desenvolvimento de software são capazes de entregar a um ritmo muito mais rápido, principalmente a adoção de métodos ágeis de desenvolvimento de software (VIRMANI, 2015), as equipes desenvolvem de forma acelerada, enquanto a equipe de operações consegue implantar de forma sequencial. Sendo assim *DevOps* nasceu a partir dessa necessidade de implantar software no mesmo ritmo em que as equipes são capazes de entregar novas funcionalidades.

DevOps pode ser visto como um conjunto de práticas e princípios para a entrega de software, onde a chave é o foco e a velocidade de entrega e automação, para pode auxiliar na capacidade de reagir a mudanças (VIRMANI, 2015). A automação da implantação de software vem principalmente pela necessidade do alinhamento entre o time de desenvolvimento com o time de operações. Essa necessidade vem em relação a ligação entre as duas equipes e ao processo de implantação de software, ferramentas para automatizar implan-

tações de software, as atividades de implantação e responsabilidades dentro do ciclo de desenvolvimento.

O que acontecia é que o time de desenvolvimento terminavam suas funcionalidades e seus testes e entregavam a uma equipe de implantação, e a equipe de implantação precisava lidar com os problemas sem a ajuda do time de desenvolvimento. Por isso *DevOps* tenta unir esses dois mundos, fazendo com que as duas equipes compartilhem atividades e responsabilidades (SPINELLIS, 2012b).

A comunidade *DevOps* defende a comunicação entre a equipe de operações e a equipe de desenvolvimento como um meio de assegurar que o os desenvolvedores entendam os problemas associados com as operações. Um dos principais benefícios disso é a capacidade de quantificar os problemas dos dois mundos, para que possa levar a melhoria do desenvolvimento do produto (HTTERMANN, 2012).

Em Virmani (2015) é abordado as práticas *DevOps*, que podem ser aplicadas diversas vezes dentro de um ciclo de desenvolvimento de software, as praticas *DevOps* podem ser resumidas em:

- **Planejamento Contínuo:** O planejamento da equipe de operações deve ser sempre contínuo e evoluído junto ao planejamento da equipe de desenvolvimento, com tarefas sendo priorizadas o tempo todo e sempre alinhadas de acordo com as decisões tomadas em conjunto com a equipe de desenvolvimento.
- **Integração Contínua:** Compartilhar as alterações feitas com toda equipe evitando que as novas modificações fiquem apenas nas mãos do time de desenvolvimento.
- **Implantação Contínua:** O coração do *DevOps*, recomenda-se automatizar todo o processo de implantação, removendo quaisquer tipo de etapas manuais com auxílio de uma ferramenta que possa automatizar a instalação, sendo assim a implantação de cada nova versão de software passa a ser de forma mais rápida e eficiente.
- **Testes Contínuos:** Automatiza também os testes da sua aplicação.
- **Monitoração Contínua:** Monitorar as novas alterações que foram implantadas a fim de aumentar a capacidade de reagir a quaisquer surpresas em tempo hábil.

Os benefícios do uso do *DevOps* são vários, como por exemplo: economia de tempo, já que implantação de software passa a ser um processo natural, economia de custos, aumentar a organização e eficiência do desenvolvimento de software como um todo (VIRMANI, 2015).

Para a entender melhor a prática de implantação automatizada, que é o coração do *DevOps*, é necessário possuir um conhecimento prévio sobre a infraestrutura como

código e as ferramentas que são utilizadas para automatizar a implantação de software, conhecendo algumas ferramentas utilizadas no mercado para automatizar implantação e suas maneiras de atuação, além das técnicas de implantações utilizadas, como por exemplo o empacotamento de software.

2.1.2.1 Infraestrutura como código

Infraestrutura como código é a prática de especificar configurações de sistema de computação através de código, com o intuito de automatizar a implantação de um software e o seu gerenciamento de configurações. Por exemplo, um sistema com diferentes configurações de hardware e diferentes requisitos de software podem ser especificados e implantados automaticamente sem intervenção humana (SHARMA; FRAGKOULIS; SPINELLIS, 2016). Tal procedimento pode ser ainda personalizado conforme as necessidades da implantação, sendo assim, a implantação automatizada não só é mais rápida do que o processo manual, mas também é mais confiável e reproduzível.

Esse conceito se complementa com o uso das práticas *DevOps*, com o objetivo de automatizar toda a sua infraestrutura, utilizando ferramentas que auxiliam nesse processo. Para aplicar a infraestrutura como código em um projeto, é necessário migrar a infraestrutura do projeto em código, essencialmente como um código que o desenvolvedor lida no seu trabalho diário (SPINELLIS, 2012a).

Dado que um script de configuração realmente contém código, é mais natural para o desenvolvedor tratá-lo como tal, utilizando suas habilidades de codificação, e poder utilizar ferramentas que auxiliem a prática do *DevOps* e de colaboração, entre a equipe de desenvolvimento e a equipe de infraestrutura (SPINELLIS, 2012a).

Com a infraestrutura sendo tratada como código, é possível utilizar um sistema de controle de versão, permitindo que outras pessoas colaborem e trabalhem em equipe. Com a ajuda de um sistema de controle de versão por exemplo, é possível ter uma documentação completa do estado do seu sistema, através do histórico de alterações, para que para que seja possível trazer o código da infraestrutura para um estado desejado a partir de qualquer outro estado.

Para realizar essa prática, existem ferramentas como Chef [1] e Puppet [2], que fornecer aos desenvolvedores várias abstrações para expressar etapas de automação, de forma independente, esses recursos podem ser executado várias vezes obtendo-se sempre o mesmo resultado, trazendo uma capacidade de atingir um determinado estado desejado sob de diferente circunstâncias, com potencial de atingir múltiplas iterações, permitindo implantações frequentes de infraestruturas complexas (HUMMER et al., 2013).

[1] https://www.chef.io/chef/
[2] https://puppet.com/

2.2 Métodos e ferramentas para implantação automatizada de software

Existem várias soluções técnicas para melhorar a implantação de software, no entanto, apesar da sua importância para as empresas de software, as atividades de implantação de software têm recebido pouca atenção (MANTYLA; VANHANEN, 2011). Recentemente um número de novas tecnologias começaram a emergir para resolver o problema de implantação. As características típicas oferecidas por estas tecnologias incluem sistemas para automatizar a implantação a partir de configurações, pacotes, gerenciamento de rede e instalação de recursos, com propósito de entrega das atualizações de forma automática (CARZANIGA et al., 1998).

Para a implantação automatizada de software é possível utilizar linguagens de script de propósito geral como: Python, Ruby e Shellscript. Também existem ferramentas gerais voltados para o processo de implantação, como sistemas de middleware especializados em determinados tipos de artefatos implantáveis (LEITE, 2014).

Um processo de implantação automatizado depende bastante da integração de diferentes papéis numa organização. Foi dito também na seção 2.1.2 que é importante a integração entre desenvolvedores e operadores, uma vez que o desenvolvimento desses scripts ou utilização das ferramentas precisam de participação de ambos os perfis.

Assim, um grande facilitador da implantação de software são as ferramentas que auxiliam na automação das atividades de implantação, pois dependendo do software, a implantação pode se tornar um processo com muitas atividades ou bastante longo, e a automação busca simplificar a implantação, para evitar erros de instalações manuais e facilitar a replicação em vários ambientes diferentes.

Tais ferramentas permitem-nos controlar e automatizar a configuração de todos os elementos que compõem um sistema, são eles: Hosts, software a serem instalados, usuários do sistema, serviços em execução, arquivos de configuração, tarefas agendadas, configuração de rede, armazenamento de arquivos, monitoramento e segurança, com sua função principal de automatizar a instalação e configuração de um sistema.

Assim é possível escrever algumas regras, que expressam como um software deve ser configurado, e assim a ferramenta configurará o sistema conforme as especificações (SPINELLIS, 2012b). Essas ferramentas funcionam de forma declarativa, especificando a instalação de um sistema de através de regras (SPINELLIS, 2012b). i

Por exemplo, é possível especificar regras para a configuração do sistema a partir de uma infraestrutura básica, no Chef essa estrutura é conhecida como livro de receitas, em Puppet essa estrutura é conhecida como manifesto, esses recursos são basicamente arquivos, no qual o implantador define os passos que serão executados, cada um podendo

executar, por exemplo: um aplicativo, um serviço, papéis de usuários do sistema, arquivos de configuração e etc.

Cada passo da configuração desejada pode depender de outros passos, ou seja, para instalar uma aplicação é preciso instalar previamente um compilador ou interpretador da linguagem, um exemplo é que para rodar uma aplicação web é necessário a instalação prévia de um servidor web. Para lidar com as dependências essas ferramentas descrevem o estado em que a aplicação deve estar, ou seja, todos os pacotes, arquivos de configuração diretórios e usuários, sendo possível definir a dependência entre as tarefas a serem executadas.

Com essas ferramentas, a implantação de um software dentro de uma infraestrutura não precisa de esforço manual, com elas podemos automatizar toda a implantação de um software, desde a preparação do ambiente até a inicialização da aplicação, sendo assim possível reduzir o tempo gasto na implantação, sem qualquer esforço adicional, o que integra os conceitos de *DevOps* com implantação de software automatizada.

Além das ferramentas de automação, é importante também entender o empacotamento de programas. Tanto Chef como Puppet utilizam a instalação de pacotes como recurso para poder instalar softwares, ou seja, é possível buscar os softwares desejados a partir dos pacotes disponíveis numa distribuição Linux. As distribuições Linux que optam por disponibilizar pacotes mantém uma infraestrutura de servidores como fonte de distribuição de programas (ARAÚJO, 2011), tais servidores são chamados de repositórios, e esses pacotes são gerenciados com softwares conhecidos como sistemas gerenciadores de pacotes, que possuem a responsabilidade de buscar os softwares a serem instalados que estão disponíveis nos respectivos repositórios.

Existem pacotes que dependem do sistema operacional, como por exemplo, pacotes .deb para a distribuição Debian e pacotes .rpm para a distribuição Fedora, e pacotes independentes de sistema operacional, como por exemplo, pacotes jar da linguagem java. Neste trabalho trataremos apenas de pacotes Debian.

2.2.1 Pacotes Debian

O formato utilizado pelo Debian GNU/Linux e seus derivados é o formato de pacotes binários conhecido como .deb, um pacote .deb é composto de arquivos executáveis, bibliotecas e documentação associada a um programa ou a um conjunto de programas, além de todos os dados e procedimentos necessários para instalar, configurar e remover aplicativos de um sistema (ARAÚJO, 2011). A estrutura dos pacotes .deb e seus requisitos para que sejam distribuídos oficialmente pelo Debian, estão especificados no Manual de Políticas do Debian (DEBIAN, 2015).

Ao instalar um pacote .deb, são feitos todos os procedimentos necessários para a

instalação de uma aplicação, porém, existem ainda casos em que o usuário precise fazer algumas configurações, um exemplo é quando o usuário precisa configurar um banco de dados ou fazer uma configuração específica de uma aplicação a partir de arquivos de configuração.

Essas tarefas não são de responsabilidade dos pacotes e sim do usuário que deseja utilizar esse software, já que o pacote não poderia prever qual é a estrutura e a configuração que o usuário pretende utilizar, logo a execução desses passos são feitas de acordo com a preferência do usuário, podendo ser feita manualmente ou a partir de uma implantação automatizada que utilize alguma ferramenta para isso. Um exemplo desse tipo de atividade, que não é de responsabilidade de um pacote, é a configuração de hospedagem virtual, que será visto na subseção a seguir.

2.2.2 Múltiplas Instâncias de Aplicações Web com Hospedagem Virtual

Na implantação de software é necessário escolher um ambiente alvo, onde a aplicação será instalada. Esse ambiente deve possuir um nó, que é um recurso computacional onde se implanta um componente. Porém pode existir a necessidade de implantar várias aplicações no mesmo servidor, e no caso de aplicações web isso pode ser um problema, visto que as aplicações estarão em um único servidor, assim também em um único endereço de *IP*.

Uma forma de solucionar esse problema é o uso de hospedagem virtual, que é um termo que se refere à prática de executar mais de uma aplicação web numa única máquina (APACHE, 2015). Com a hospedagem virtual é possível hospedar mais de um nome de domínio no mesmo computador, o que possibilita ter vários nomes de domínio em execução, num único endereço de *IP*. Isso possibilita que, num mesmo servidor, tenha uma aplicação com o endereço www.blog.com e outra com o endereço www.nuvem.com.

O fato de que eles estão em execução no mesmo servidor físico não é aparente para o usuário final. Essa solução é importante para economizar recursos, já que não será necessário possuir um servidor dedicado apenas a uma aplicação.

Alguns softwares possuem suporte para a implementação de hospedagem virtual, como por exemplo o Apache e o Nginx, o funcionamento de ambos é a partir de arquivos que especificam a configuração da hospedagem virtual e determina como o servidor irá responder às várias requisições de domínio. No apache o arquivo padrão de configuração é chamado 000-default.conf que pode ser utilizado como referência, como por exemplo:

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

[caption= Exemplo de arquivo de configuração de hospedagem virtual no apache]

Isso quer dizer que as requisições na porta 80 vindas de webmaster@localhost serão direcionadas aos arquivos que estão em /var/www/html. Já no Nginx a configuração de hospedagem virtual também é feita através de arquivos de configuração, o arquivo padrão no Nginx é chamado de default.conf que também pode ser utilizado como referência, como por exemplo:

Código 2.1 – Exemplo de arquivo de configuração de hospedagem virtual no Nginx

```
server {
        listen    80;
        root /var/www/example.com/public_html;
        index index.html index.htm;

        # Make site accessible from http://localhost/
        server_name example.com;
}
```

Isso quer dizer que as requisições que são ouvidas da porta 80 serão direcionadas aos arquivos que estão em /var/www/example.com/public_html, também é indicado o arquivo index que será o arquivo da página inicial do nome do servidor na porta 80, além disso é definido example.com para o nome servidor. Além disso existem vários outros recursos que essas ferramentas possuem, como por exemplo o recurso de proxy, porém para este trabalho será necessário apenas o uso de hospedagem virtual.

2.3 Segurança na Implantação de Aplicações Web

Como vimos, um pacote cumpre bem suas responsabilidades dentro de uma implantação de software, porém outras atividades também são importantes e que fogem da responsabilidade de um pacote, outro fator importante é que um pacote também não pode prever é quais são os procedimentos de segurança que devem ser tomados numa implantação, como por exemplo a criação de certificados digitais. Para entender quais são esses procedimentos, será abordado quais são os procedimentos de segurança na implantação de aplicações web utilizando pacotes Debian, no qual trata o uso de protocolos que utilizam criptografia para a segurança de dados, que também são procedimentos que podem ser feitos manualmente ou utilizando alguma ferramentas para automatizar o processo.

De acordo com (KUROSE et al., 2010) existem algumas categorias predominantes e prejudiciais de ataques na internet, são eles: ataques via malware, recusa de serviços,

analisador de pacotes, disfarce na fonte e modificação e exclusão de mensagem, e para proteger desses ataques é importante o uso de comunicação segura, protegendo as aplicações. Para isso Kurose ainda identifica as propriedades desejáveis da comunicação segura, são elas:

- **Confidencialidade:** Diz respeito a somente o remetente e o destinatário pretendido podem entender o conteúdo da mensagem transmitida.
- **Autenticação do ponto final:** Diz respeito a tanto o remetente como o destinatário precisam confirmar a identidade da outra parte envolvida na comunicação.
- **Integridade da Mensagem:** Diz respeito a assegurar que o conteúdo da mensagem não seja modificado.
- **Segurança Operacional:** Diz respeito a segurança da rede, no qual os atacantes podem tentar adquirir os segredos da rede e lançar ataques DoS, por isso é necessário uma boa configuração de firewall e sistemas de detecção de invasão.

No contexto implantação de aplicações web, tomaremos como objetivo a segurança dos dados transmitidos, com isso aplicando criptografia nos protocolos da camada de aplicação, que serão utilizados neste trabalho, sendo eles os protocolos *HTTP* e *SMTP*, assim adicionando uma camada de segurança as aplicações que serão implantadas.

2.3.0.1 Protocolos da Camada de Aplicação e Camada de Segurança

As aplicações web utilizam protocolos da camada de aplicação, que é uma das cinco camadas do modelo utilizado por Kurose em (KUROSE et al., 2010), a camada de aplicação é onde residem as aplicações de rede e seus protocolos, incluindo os protocolos *HTTP* e *SMTP*, que são protocolos importantes neste trabalho, já que aplicações web utilizam o *HTTP* para transferência de arquivos e o *SMTP* para transferência de mensagens de correio eletrônico. Os protocolos *HTTP* e *SMTP* estão especificados em RFCs, a especificação do protocolo *HTTP* está disponível na RFC 2616 e RFC 1945, já o protocolo *SMTP* está especificado na RFC 5321.

A aplicações que serão utilizadas neste trabalho são aplicações web empacotadas no Debian, e aplicações web costumam ser aplicações cliente-servidor, que permitem aos usuários obterem documentos de servidores web a partir de requisições (KUROSE et al., 2010), esses documentos são padronizados, como por exemplo o *HTML*, para que os navegadores possam interpretar e passar a informação ao usuário.

De acordo com (KUROSE et al., 2010) o *HTTP* é um protocolo da camada de aplicação da web, e é implementado em dois programas, um cliente e um servidor, conversando um com outro a partir de troca de mensagens, o papel do *HTTP* é definir a

estrutura dessas mensagens e o modo como o servidor e o cliente as trocam. O principal conteúdo das mensagens trocadas entre clientes e servidores são os objetos, um objeto é basicamente um arquivo, podendo ser um vídeo, uma imagem ou um arquivo *HTML*.

Já o *SMTP* é protocolo responsável por transferir mensagens de servidores de correio remetentes para servidores de correio destinatários, também transferindo arquivos, de um servidor de correio para o outro. Uma diferença entre o *HTTP* e o *SMTP* é que o protocolo *HTTP* é um protocolo de recuperação de informações, enquanto o protocolo *SMTP* é um protocolo de envio de informações (KUROSE et al., 2010). Isso implica que para um usuário recuperar as informações de e-mail é necessário o uso de outros componentes, já que isso não é possível via *SMTP*, para resolver isto existem os protocolos de acesso ao correio, entre eles o *POP3* e *IMAP*.

Para adicionar uma camada de segurança, com sigilo e integridade de dados é necessário o uso de outro protocolo, o protocolo *SSL*, para complementar os protocolos *HTTP* e *SMTP*. O protocolo *SSL* é utilizado por basicamente todos os sites conhecidos, como Google, Amazon, eBay, e seu uso é para oferecer segurança em transações, pois ele cria um canal criptografado entre um servidor web e um navegador, garantido que todos os dados transmitidos sejam sigilosos e seguros (KUROSE et al., 2010). Para poder utilizar o *SSL* é necessário que o seu servidor web possua um certificado, que possa responder algumas questões sobre a identidade do seu servidor.

Com as devidas configurações, tanto *HTTP* como *SMTP* estarão sobre uma camada adicional de segurança, que utiliza o protocolo *SSL*/TLS. Essa camada adicional permite que os dados sejam transmitidos por meio de uma conexão criptografada e que se verifique a autenticidade do servidor e do cliente por meio de certificados digitais. Agora o próximo passo é ver o que são os certificados digitais e seu papel dentro da segurança em aplicações web.

2.3.0.2 Assinaturas e Certificados Digitais

As assinaturas e certificados digitais servem para agregar confiança e segurança nas trocas de dados pela internet, principalmente quando se trata de troca de informações sigilosas e que precisam de uma confiabilidade maior, basta imaginar que ninguém gostaria de ter seus dados bancários expostos, ou ter seus dados trafegando sem nenhum tipo de segurança numa rede aberta. De acordo com (KUROSE et al., 2010) assinatura digital é uma técnica criptográfica usada para atestar que você e não outra pessoa conhece e ou concorda com tal tipo de documento. O que viabiliza a assinatura digital é o emprego de criptografia assimétrica ou criptografia de chaves públicas, logo a assinatura digital deve ser algo verificável e não falsificável. Uma aplicação importante de assinaturas digitais é a certificação de chaves públicas, que certifica que uma chave pública pertence a uma entidade específica(KUROSE et al., 2010), e a certificação de chaves públicas é utilizada

no protocolo *SSL*, visto anteriormente.

Uma maneira de se obter o certificado, é através de uma entidade conhecida como entidade certificadora, que é responsável por validar e emitir certificados, e verifica se uma entidade é realmente quem ela diz ser, após essa validação a entidade certificadora cria um certificado que vincula a chave pública da entidade a essa verificação, assim o certificado passa a ter a chave pública e a identificação de que é realmente o proprietário daquela chave pública.

Porém essas entidades certificadoras cobram por esse serviço, e os certificados também possuem validade. Uma outra maneira de se obter certificados é a partir dos certificados autoassinados, que são os certificados gerados por conta própria através de ferramentas, um exemplo de ferramenta é a ferramenta OpenSSL, que permite requisitar, assinar, gerar, exportar e converter certificados digitais.

Um problema em utilizar certificados autoassinados é que os clientes, como navegadores, precisam apenas dos certificados das entidades certificadores em quem eles confiam, por exemplo, um navegador não vai conhecer o certificado que foi autoassinado por alguém que ele não conhece, e logo vai avisar ao usuário de que ele não está numa conexão segura, por mais que esteja sobre uma conexão *HTTP*. Por isso que manter certificados autoassinados pode tornar-se mais difícil, porém é uma saída principalmente para fins de testes, e melhor do que não utilizar nenhum tipo de segurança.

Entretanto, em 2014 a universidade de Michigan, e empresas como a Mozilla, criaram uma nova entidade certificadora, que se chama Let's Encrypt, que fornece certificados digitais de forma gratuita e que são reconhecidos por todos os navegadores, utilizando o protocolo *ACME* (INMETRO, 2015), existe um cliente ACME configurado no servidor do domínio, que será consultado pelo servidor da autoridade certificadora, validando assim o domínio em questão, em resumo, let's encrypt faz o trabalho de automatizar a validação da identidade pela autoridade certificadora.

3 Definição e Preparação dos Estudos

Este capítulo aborda sobre todo o planejamento para a execução do projeto, contendo os procedimentos e técnicas utilizados, a fim de explicar como o desenvolvimento foi realizado para atingir os seus objetivos, servindo como base para a sua reprodução em trabalhos futuros.

O trabalho é fundamentado na construção de uma solução utilizando conceitos e práticas da engenharia de software, com a definição de um objetivo, a definição de um panejamento, e a execução das atividades planejadas, e posteriormente a validação dos resultados. O objetivo deste trabalho é resolver a seguinte questão:

Como implantar aplicações web em sistema Debian GNU/Linux de forma automatizada e e segura?

Dado objetivo e a questão problema, este trabalho consiste em contribuir com a construção de uma solução de engenharia de software que permita implantar aplicações web em sistemas Debian GNU/Linux de forma automatizada e segura, essa construção será feita a partir dos conhecimentos adquiridos durante o curso de engenharia de software, além disso a solução proposta deverá passar por uma validação dos resultados, baseada na observação da execução da solução em exemplos de uso, onde cada exemplo conterá uma aplicação real que deverá ser implantada com sucesso. Essa validação tem seus procedimentos definidos na seção 4 e as definições dos exemplos de uso estão na seção 3.4.1.

Inicialmente será feito uma pesquisa dos trabalhos relacionados para compreender quais são as soluções na engenharia de software que buscam a automação da implantação de aplicações web, essa pesquisa é uma pesquisa aplicada pois tem como objetivo gerar conhecimentos para aplicação prática (GERHARDT; SILVEIRA, 2009), relacionada a aplicações que auxiliam na implantação automatizada de software.

A partir dessa pesquisa será possível encontrar os trabalhos relacionados, os trabalhos encontrados também serão de insumo para auxiliar na construção da solução a partir da análise dos resultados que foram encontrados. As próximas seções tratam dos trabalhos relacionados encontrados e dos métodos e procedimentos para construção da solução, além da validação da solução.

3.1 Trabalhos Relacionados

Para responder a questão problema primeiramente deve ser feito um levantamento de trabalhos relacionados, para compreender como a automação da implantação de software está sendo utilizada como solução na academia e nas empresas, e quais são os procedimentos e técnicas utilizados.

Para entender o que acontece na comunidade da computação em relação a implantação automatizada de aplicações, foi feito uma busca de alguns trabalhos relacionados. O primeiro trabalho relacionado é um trabalho acadêmico feito por (LEITE, 2014) no qual foi desenvolvido um sistema de middleware chamado CHOReOS Enactment Engine que é um sistema que possibilita a implantação distribuída e automatizada de composições de serviços web numa estrutura virtualizada, no qual opera no modelo computacional conhecido como plataforma como serviço, comparando esse sistema com abordagens tradicionais de implantação levando em consideração a escalabilidade em relação ao tempo de implantação das composições dos serviços. Seu foco maior é na automação da implantação automatizada de aplicações aliado com a gerência de recursos de hardware.

Existem também ferramentas disponíveis no mercado que também trazem a proposta de automação de instalação de aplicações alguns exemplos são: (BITNAMI, 2015) BitNami que é uma biblioteca de aplicativos populares e ambientes de desenvolvimento que podem ser instalados com apenas um clique, através de uma interface web amigável. Ele automatiza todo o processo de compilar e configurar os aplicativos e todas as suas dependências (bibliotecas de terceiros, linguagem de programação, bases de dados) para que o usuário comum não se preocupe com questões técnicas. Também temos o (SANDSTORM.IO, 2015) Sandstorm é uma plataforma de código aberto para servidores pessoais. Sandstorm permite que você execute o seu próprio servidor e instalar facilmente as aplicações em que o próprio Sandstorm dá suporte, e por fim a ferramenta (JUJU, 2015) Juju que lhe permite implementar, configurar, gerenciar, e manter serviços de forma rápida e eficiente automatizando a instalação e configuração de aplicações na nuvem.

O estudo dessas aplicações serve como base para entender como funcionam as aplicações já existentes, e como elas são feitas, para que assim seja possível identificar as ferramentas que são utilizadas para solucionar problemas semelhantes ao problema deste trabalho, por exemplo a ferramenta JuJu utiliza uma abstração conhecida como charm, que é basicamente um arquivo com trechos de código, esse código define um serviço, como por exemplo: instale um banco de dados ou instale a aplicação Wordpress. Um charm contém toda a lógica de que você precisa para implementar e integrar uma aplicação, contendo todo o processo de download de ferramentas, instalação e configuração de aplicações, podendo ser por provisionadores como Chef, Puppet ou Docker (JUJU, 2015), é possível reaproveitar os charms feitos pela comunidade do JuJu assim reaproveitando os charms prontos ou não precisando escrever um charm do zero.

Já Bitnami já trás as aplicações prontas pra uso, assim o usuário interage com uma interface web, no formato clique para instalar, porém o usuário só tem a disposição as aplicações disponibilizadas pelo Bitnami, isso também é a forma em que o Sandstorm trabalha, com a diferença de que o Sandstorm é software livre, já Bitnami tem o código fechado, o que dificulta a análise da arquitetura que é utilizada por eles. Por fim o trabalho feito por (LEITE, 2014) é voltado para serviços web de grande escala, com foco em implantação de aplicações na nuvem e escalar infraestrutura, utilizando o middleware construído em seu trabalho, assim podendo gerenciar ambientes de computação em nuvem como Amazon EC2 e Openstack, a sua aplicação utiliza o Chef solo como seu agente de configuração.

Com esse levantamento, as soluções encontradas não resolvem totalmente o problema, a grande diferença é que as ferramentas encontradas não utilizam os pacotes disponibilizados pelo Debian, mas elas servem como motivação para o desenvolvimento deste trabalho. As ferramentas encontradas buscam abstrair todo processo de instalação e configuração das aplicações, para que o usuário não tenha dificuldades na instalação de aplicações web, as ferramentas também trazem uma interface web para que o usuário não precise executar tarefas via terminal.

Outra informação importante é que as ferramentas também utilizam do conceito de infraestrutura como código, como visto no capítulo 2, e para isso utilizam provisionadores como Chef ou Puppet, tornando mais simples a construção de códigos para a instalação e configuração de aplicações. Tendo em vista que pelo levantamento dos trabalhos relacionados, o problema inicialmente proposto não foi resolvido, justifica-se a construção de uma solução de engenharia de software para implantação automatizada de aplicações em sistemas Debian GNU/Linux.

3.2 Construção da solução

A partir da análise encontrada nos trabalhos relacionados, é necessário definir como será feito a construção da solução, primeiramente será definido as ferramentas de apoio ao desenvolvimento da solução, após isso, a escolha da ferramenta que será utilizada para solução do problema, essa será a ferramenta que será evoluída durante este trabalho. Após a escolha das ferramentas, será descrito como será feito a validação da solução nos exemplos de uso com aplicações web escolhidas que possuem seus pacotes disponibilizados pelo Debian.

3.2.1 Ferramentas de apoio

As ferramentas de apoio ao desenvolvimento de software são importantes para a organização do trabalho, auxiliando nas atividades típicas dentro de um projeto de

engenharia de software. Algumas ferramentas são importantes pois tornam mais fácil a execução de algumas atividades, como por exemplo: o controle de versão do software, ferramenta para documentação do software e ferramenta para gerenciar as tarefas. Os critérios para escolha das ferramentas são:

- **Aplicação para sistema de controle de versão git:** Uma ferramenta que possa gerenciar diversas versões do desenvolvimento do código fonte utilizando o sistema de controle de versão git, que seja de preferência um software livre ou que não cobre o serviço de hospedagem. Algumas ferramentas disponíveis para isso são Github, Gitlab e Bitbucket.
- **Ferramenta para documentação de código:** Uma ferramenta que possa documentar o projeto, as aplicações como Gitlab e Github possuem uma wiki já disponível para o projeto.
- **Ferramenta de gerenciamento de tarefas:** Uma ferramenta que possa gerenciar as tarefas que serão executadas, que estão em execução ou que vão ser executadas durante o desenvolvimento do projeto. Essa ferramenta será importante para a organização do trabalho, dando visibilidade do trabalho que está sendo realizado.
- **Ferramenta de comunicação:** A ferramenta de comunicação será importante para tirar dúvidas rápidas em relação a dificuldades e desafios, as comunidades de software livre costumam usar listas de e-mail e canais no *IRC* para a comunicação de seus desenvolvedores e usuários.

As ferramentas escolhidas para o apoio ao desenvolvimento foram:

- **Ferramenta Git:** O Gitlab foi a aplicação escolhido para o armazenar o repositório git, é semelhante ao Github, porém é software livre, distribuído pela licença MIT(ENTERPRISE, 2015).
- **Documentação de Código:** Toda documentação será disponibilizada na wiki do projeto no Gitlab.
- **Gerenciamento de Tarefas:** O gerenciamento de tarefas também será feito pelo projeto do Gitlab, onde é possível criar atividades para serem feitas e criar marcos podendo adicionar as atividades a esses marcos. Como o ciclo de desenvolvimento será de uma semana, serão criados marcos para cada semana e atividades agrupadas em cada marco.
- **Comunicação:** A comunicação será feita via *IRC*, que é uma ferramenta de comunicação no formato de chat com canais e usuários dentro dos canais, apesar de

ser uma ferramenta antiga ainda é muito usada pelas comunidades de software livre e grandes empresas como Facebook (FEITELSON; FRACHTENBERG; BECK, 2013).

3.2.2 Ferramenta Shak

De acordo com a motivação em contribuir com projetos de software livre e a participação do google Summer of Code, como dito na seção de motivação **??**, e todo o contexto definido na seção 3.2, a construção da solução proposta neste trabalho será baseada na evolução da ferramenta Shak (Self Hosting Applications Kit).

A ferramenta Shak tem o objetivo de facilitar ao máximo que usuários sem conhecimento técnico possam ter os seus próprios serviços de internet, garantindo a sua privacidade e segurança. Esta plataforma está concebida de acordo com os seguintes princípios:

Base: Pacotes Debian

A plataforma Shak é baseada no sistema de pacotes do Debian, uma das distribuições GNU/Linux mais consolidadas e com maior comunidade internacional. Os pacotes do Debian fornecem atualizações consistentes de correção e de segurança, e são utilizados por uma grande quantidade de usuários que reportam problemas a serem resolvidos. Porém, para garantirem uma flexibilidade e suportar diferentes casos de uso, normalmente os pacotes não realizam a configuração completa do ambiente, deixando os detalhes finais para o administrador do sistema.

Além disso, normalmente as aplicações web são compostas de diversos pacotes, e requerem configurações específicas que dizem respeito a mais de uma aplicação, e que por consequência não poderiam ser feitas automaticamente de uma forma sustentável em nenhum dos pacotes. Por isso, os pacotes além de serem instalados, precisam ser configurados, e isso é feito muitas vezes de forma manual, ou ainda, requer conhecimento técnico para executar configurações mais complexas, afastando os possíveis usuários que queiram utilizar alguma aplicação fornecida pelo Debian, mas não possuem conhecimento técnico para a configuração das aplicações.

Nova abstração: Aplicação

Para resolver as questões que não podem ser resolvidas a nível de pacotes, o Shak introduz uma nova abstração: a aplicação. Uma aplicação geralmente contém mais de um pacote, inclusive necessita de configurações em vários pacotes.

Uma vez que o usuário seleciona uma determinada aplicação para ser instalada, os pacotes necessários são instalados e as configurações necessárias são feitas de forma automática, fornecendo de fato um "instalador de um clique"para aplicações suportadas.

Todas as configurações são feitas de forma consistente utilizando uma ferramenta comum, buscando o maior nível de segurança possível, e assumindo responsabilidades que não podem ser tomadas por um pacote, e realizando tarefas complexas de forma automatizada, sem a necessidade da interferência de algum usuário.

Diferencial para outras soluções existentes

Existem outras soluções disponíveis para instalação de aplicações por um clique, como por exemplo (BITNAMI, 2015) e (SANDSTORM.IO, 2015), mas o Shak se diferencia delas nos seguintes aspectos:

- O Shak pode ser implantando numa máquina virtual na nuvem ou em servidores físicos próprios, a critério do usuário.
- O Shak reutiliza o trabalho dos mantenedores Debian, que fornecem pacotes de alta qualidade.
- O Shak reutiliza a infraestrutura do Debian, que é utilizada por várias outras distribuições Linux baseadas no Debian, como por exemplo: Ubuntu e Linux Mint.

Por outro lado, apenas os softwares que estejam empacotados no Debian estarão disponíveis, mas isso pode ser visto de forma positiva também: o Shak fornecerá um incentivo para que ainda mais softwares estejam disponíveis no Debian, o que é um benefício coletivo para o ecossistema do software livre, e para os seus usuários. Logo, quanto mais aplicações disponíveis no Debian, mais aplicações serão disponibilizadas para os usuários do Debian e seus derivados.

O Debian possui espelhos do seu repositório espalhados por todo o mundo, fornecendo links para downloads mais rápidos aonde quer que o usuário esteja. Atualizações de defeitos e correções de segurança realizadas no Debian estarão automaticamente disponíveis para usuários do Shak de forma transparente. Além disso, o próprio Shak estará disponível como um pacote dentro do próprio repositório oficial do Debian.

A arquitetura da ferramenta Shak é composta de:

- **Livro de Receitas Chef:** Shak contém alguns livros de receitas para poder organizar a instalação de cada componente, como por, exemplo um livro de receitas para cada aplicação que for instalada.
- **Código Ruby** A arquitetura do Shak foi desenvolvida na linguagem Ruby, com programação orientado a objetos.
- **Servidor web** O Shak utiliza o software livre Nginx, que é servidor *HTTP* de alto desempenho (NGINX, 2015).

- **Pacotes Debian** O Shak utiliza pacotes incluídos na distribuição oficial do Debian.
- **Gems** O Shak também utiliza algumas gems para seu funcionamento, uma gem nada mais é do que uma biblioteca Ruby, que provê um formato padrão para a distribuição de programas Ruby (RUBYGEMS, 2015).
- **Código Shellscript** O Shak também utiliza algumas gems para seu funcionamento, uma gem nada mais é do que uma biblioteca, que provê um formato padrão para a distribuição de programas Ruby (RUBYGEMS, 2015).

O Shak é uma ferramenta que põe em prática o conceito de infraestrutura como código, buscando velocidade e principalmente qualidade na implantação automatizada de aplicações, utilizando-se de codificação simples e objetiva, sem a necessidade mais de diversos passos e processos para se preparar um ambiente, e sem perder o poder de controle, segurança e qualidade da implantação das aplicações.

Para adicionar no Shak uma aplicação que esteja disponível nos repositórios oficiais do Debian, é necessário construir o livro de receitas para a aplicação desejada, essa atividade será uma das atividades deste trabalho, sendo assim adicionando novas aplicações web a ferramenta Shak.

3.2.3 Criação de Ambientes de Desenvolvimento

Uma das necessidades para o desenvolvimento do projeto é ter um ambiente de desenvolvimento flexível que possa ser rapidamente construído e destruído, onde possa testar as implantações feitas pela ferramenta Shak e verificar os erros caso aconteçam. Para isso é necessário utilizar alguma ferramenta que automatize processo de construir ambientes de desenvolvimento, construir ambientes manualmente pode ser um processo longo e demorado, além disso evitar que ocorra erros por desatenção ao executar vários passos manuais, assim para automatizar a implantação das aplicações é necessário o uso de uma ferramenta para auxiliar na criação das máquinas virtuais, onde é possível gerenciá-las de forma automatizada, servindo de suporte para a criação dos ambientes de desenvolvimento e facilitando a execução e teste da implantação das aplicações escolhidas.

A ferramenta escolhida para auxiliar a criação de um ambiente de desenvolvimento é a ferramenta Vagrant, com o Vagrant é possível gerenciar a criação de máquinas virtuais para os ambientes de desenvolvimento do projeto, também é possível usar ferramentas de automação para criar o servidor de implantação, definindo todas as configurações necessárias em arquivos que farão parte do repositório git do Shak. Dessa maneira, qualquer pessoa que clonar o repositório poderá rodar uma instância desse servidor, que será exatamente idêntico para todos os usuários que forem criar uma máquina virtual, reduzindo

ao máximo os problemas que surgem por causa das diferenças entre sistemas operacionais e configurações de sistema.

Quando executamos o comando vagrant up o Vagrant cria uma máquina virtual com as configurações que estão no arquivo Vagrantfile na raiz do projeto Shak, nele estão todas as configurações e definições da máquina virtual em questão. Para o Vagrant criar uma máquina virtual era necessário escolher um software que gerencie esse procedimento, a ferramenta escolhida foi o Virtualbox para poder emular o sistema operacional uma máquina virtual, assim o Vagrant inicia uma máquina virtual utilizando o Virtualbox, com a distribuição Debian GNU/Linux e as configurações que estiverem definida no arquivo de configuração Vagrantfile.

Para executar o Vagrant basta instalar o pacote disponível pelo Debian, além disso o repositório do projeto Shak já possui um Vagrantfile padrão para que todos os desenvolvedores tenham o mesmo ambiente, passa acessar a máquina virtual criada basta acessá-la via ssh, o Vagrant já gerencia as chaves ssh, portanto basta dar o comando vagrant ssh que estará conectado a máquia virtual via ssh. O ambiente de desenvolvimento levantado é um ambiente com Debian na sua versão sid 64 bits(que é a versão que contém os pacotes mais atuais, conhecida como versão instável), apesar de a versão ser a versão instável isso é importante para que possamos utilizar as versões mais novas dos pacotes das ferramentas escolhidas, também será importante para reportar algum erro caso seja encontrado.

Com o ambiente de desenvolvimento definido, agora é possível iniciar a criação dos livros de receitas das aplicações, o repositório do Shak possui alguns scripts para instalar algumas dependências ao iniciar uma máquina virtual, dentro desses scripts contém as instalações das dependências para o Shak funcionar perfeitamente e também o próprio Chef, que é utilizado pela ferramenta Shak para instalar aplicações e suas dependências, porém todos esses passos já são executados graças ao Vagrantfile que está no repositório, facilitando assim a criação de ambientes para o desenvolvimento, se a necessidade de execução de nenhum script adicional.

3.3 Planejamento das Atividades

As atividades planejadas para a evolução da ferramenta Shak são baseadas nas aplicações que foram escolhidas na seção 3.4.1 essas aplicações serão as aplicações piloto, que ajudarão na evolução da solução proposta, o planejamento inicial é que a ferramenta Shak possa concluir com sucesso a instalação do Owncloud e Wordpress de forma automatizada, os critérios para a escolha das ferramentas está definido na seção **??**, levando em consideração todos os aspectos definidos no capítulo 3. As atividades iniciais levantadas foram:

- **Atividade 1** Suporte a instalação automatizada do Wordpress.
- **Atividade 2** Suporte a instalação automatizada do Owncloud.
- **Atividade 3** Forçar as aplicações Wordpress e Owncloud a utilizar o protocolo HTTPS.
- **Atividade 4** Suporte a instalação automatizada do servidor de e-mail,
- **Atividade 5** Suporte a instalação automatizada do outras aplicações.

Ao verificar que tanto a ferramenta Owncloud como a ferramenta Wordpress precisam de configurações servidor de e-mail para algumas funcionalidades, também foi adicionado a atividade de criação de uma receita para a automatizar a configuração de servidor de e-mail. Como aspecto de segurança, ficou definido que tanto a aplicação Owncloud como Wordpress deveriam usar sempre HTTPS por padrão, isso envolve também uma estratégia de como serão gerenciados os certificados de segurança para aplicar o protocolo HTTPS. Além dessas atividades previstas, também foram feitas evoluções no Shak para que seja possível a implementação dessas atividades.

Dentro da implantação automatizada de software Debian GNU/Linux temos várias configurações possíveis, porém neste trabalho existem alguns aspectos que deverão ser levados em consideração, esses aspectos serão importantes para guiar a construção da solução.

- **Aspecto 1:** Uso dos aspectos de segurança na implantação de aplicações web, como por exemplo utilizar protocolos como HTTPS.
- **Aspecto 2:** Uso da implantação automatizada de múltiplas instâncias de aplicações no mesmo servidor, utilizando hospedagem virtual.
- **Aspecto 3:** Uso de aplicações que são empacotadas no Debian.

Por fim é necessário definir procedimentos que serão feitos para a execução da implantação automatizada das aplicações, esses procedimentos são importantes para a construção da arquitetura inicial, considerando o processo de implantação de software visto no capítulo 2 e a referência dos trabalhos relacionados. É preciso definir as fases e os procedimentos para implantação automatizada, definindo as etapas da implantação.

3.3.1 Fases e Procedimentos para implantação

Primeiramente é necessário definir as fases e os procedimentos para a implantação automatizada, seguindo as fases que compõem o processo de implantação de aplicações e

de acordo com (OMG, 2006), e isso será o que o Shak automatizará fases e procedimentos são:

- **Planejamento:** O planejamento da implantação é uma fase para identificar os componentes necessários na implantação da aplicação. Definir quais são as dependências mínimas para o funcionamento de uma aplicação, tais como: banco de dados, pacotes pré-instalados e aplicações pré-configuradas
- **Preparação e Instalação de Pacotes:** São os procedimentos necessários para preparar o ambiente alvo para que a aplicação possa ser executada, isso envolve configuração do sistema operacional, instalação e configuração de dependências necessárias, e a transferência do componente para o servidor onde ele será executado.
- **Configuração:** Como levantado no planejamento é necessário a edição de arquivos de configuração de cada aplicação, arquivos de configuração do banco de dados e arquivo de configuração do servidor web.
- **Configuração de múltiplas instâncias:** Habilitar a configuração de múltiplas instâncias de uma aplicação, permitindo que seja possível que tenha várias instâncias da mesma aplicação sem duplicação de recursos, ou seja, utilizando os mesmos recursos de um único servidor.
- **Inicialização/Execução:** Executar a instalação da aplicação pela ferramenta Shak e validar os resultados obtidos.

Também é importante definir os procedimentos de segurança na implantação, e automatizar os que forem possíveis de serem aplicados dentro do contexto da arquitetura proposta. Um exemplo é a configuração das aplicações sempre usarem protocolos seguros, como o HTTPS que possui uma camada adicional de segurança que utilizando o protocolo SSL/TLS. Para as aplicações web é necessário utilizar o protocolo HTTPS, para aplicações de e-mail é necessário utilizar os protocolos SMTP e IMAPS.

3.4 Validação da solução

Para validar a arquitetura proposta no trabalho serão feitos exemplos de uso com aplicações que possam servir para a execução da arquitetura construída, ou seja, aplicações que tenham todo o seu processo de instalação e configuração automatizado, a fim de refinar e evoluir a solução conforme problemas forem surgindo, a escolha desses exemplos de uso devem ser feitas a partir de aplicações reais e conhecidas na comunidade de software livre. Outros aspectos também são importantes e devem ser levados em consideração para a escolha das aplicações piloto, as seguintes características foram levadas em consideração para a escolha das aplicações:

- **Aplicações empacotadas no Debian:** Como o intuito do trabalho é realizar implantações múltiplas a partir de um pacote único, tais aplicações devem estar empacotadas e disponíveis para instalação nos servidores do Debian. Isso impacta na escolha da ferramenta, visto que não será necessário ter o trabalho de empacotar aplicações que ainda não estão empacotadas no Debian.

- **Servidor web compatível:** As ferramentas escolhidas devem no mínimo possuir o servidor web compatível, por exemplo: as aplicações juntas devem possuir suporte para o servidor Nginx ou Apache ou similares.

- **Aplicações com comunidades ativas:** Como estamos trabalhando com software livre, é importante que os softwares escolhidos possuem comunidades ativas, isso pode ajudar na resolução de possíveis problemas, logo aplicações em que sejam difíceis de comunicar com a sua comunidade devem ser evitadas, e aplicações com a comunidade de desenvolvedores e usuários ativa devem ser priorizadas. E isso também pode ser um fator importante caso durante os testes sejam descobertos erros ou melhorias dentro dessas ferramentas e tais erros e melhorias possam ser reportados para a comunidade, ou até mesmo problemas solucionados e devolvidos aos mantenedores das ferramentas.

- **Documentação do software:** A documentação do software também deve ser levado em consideração, principalmente a documentação da instalação e configuração dentro da ferramenta, ferramentas que não possuem documentação de instalação e configuração devem ser evitadas.

- **Aplicações com suporte a federação:** Aplicações federadas permitem utilização de métodos de autenticação única para cada instância da aplicação, mantendo a compatibilidade entre elas, um exemplo é a aplicação Owncloud, que permite compartilhar seus arquivos na nuvem independente do Owncloud que estiver usando, isto é possível utilizando apenas um identificador único, chamado ID Federated Cloud, assim possibilitando que os arquivos do usuário sejam compartilhados entre suas instâncias do Owncloud na nuvem.

A partir dessas características definidas, é necessário encontrar os exemplos de uso que possam servir para a execução da solução, todos as aplicações que servirão como exemplos de uso devem possuir pacotes incluídos na distribuição oficial do Debian.

3.4.1 Exemplos de uso

Para encontrar as aplicações que possam se encaixar dentro desses parâmetros devemos buscar por alguns exemplos de uso, para a escolha das aplicações que serão utilizadas como exemplos de uso é necessário fazer uma busca nas aplicações web que são

empacotadas no Debian e que possuem suporte a configuração de múltiplas instâncias, essa busca deve levar em consideração também a documentação para realizar tal configuração. Foram levantados algumas aplicações web empacotadas no Debian da seguinte forma:

```
apt-cache search web | wc -l
```

O resultado obtido com pacotes que contenham a palavra web recebe o resultado de 3470 pacotes de diversas aplicações ou módulos de aplicações, como: Wordpress, Owncloud, Drupal, Mailman, Chromium, etc. Logo dentro dessa rápida busca encontramos alguns pacotes de aplicações conhecidas, para facilitar a busca basta aplicar para alguns nomes de aplicações conhecidos como:

```
apt-cache search web | grep wordpress
```

Para avaliar se as aplicações possuem suporte a múltiplas instâncias é necessário analisar a documentação das aplicações, as aplicações costumam disponibilizar a sua documentação na sua página ou numa wiki, também é possível checar a documentação quando se instala uma aplicação nos arquivos de documentação da aplicação em /usr/share/doc/ ou também pode-se utilizar os comandos:

```
man "nome aplicação"
info "nome aplicação"
```

Analisando a documentação de algumas aplicações como Wordpress, Redmine, Owncloud, Mailman, Wikimédia, foram escolhidas as aplicações Wordpress e Owncloud por conter as características definidas anteriormente. Primeiro foi feito uma análise da documentação dessas ferramentas para encontrar a possibilidade da configuração de múltiplas instâncias no mesmo servidor, além disso também foi visto se as aplicações tinham suporte ao servidor Nginx que é o servidor utilizado pelo Shak, por fim as duas aplicações também foram escolhidas pela grande comunidade de usuários ativa e as duas aplicações também possuírem canais de *IRC* para tirar dúvidas com desenvolvedores e usuários, também foi levado em consideração o fato da aplicação Owncloud dar suporte a federação, com o objetivo de compartilhar arquivos em servidores na nuvem de forma simples, como por exemplo: enviar um e-mail de um servidor para outro, sendo assim, também é possível migrar os seus arquivos pessoais de um servidor Owncloud a outro.

Com as aplicações que serão exemplos de uso escolhidas a proposta de arquitetura pode ser testada e validada, com o objetivo de criar múltiplas instâncias no mesmo servidor de forma automatizada a partir de um pacote único, logo os exemplos de uso também servirão para refinar e evoluir a arquitetura proposta. Para realizar tais validações é

importante que as instalações e configurações sejam num ambiente limpo, ou seja, os testes devem ser criados de preferência em máquinas virtuais com a configuração conhecida como minimal, que contém instalado apenas as aplicações necessárias para o funcionamento do sistema operacional.

A partir da execução dos exemplos de uso será possível a coleta de informações necessárias para a validação da implantação correta das aplicações, essa validação a princípio será feita a partir da execução da implantação e verificação das funcionalidades da aplicação implantada, ou seja, após a execução da implantação o testador deverá verificar o perfeito funcionamento de algumas funcionalidades básicas da aplicação escolhida, e principalmente verificar a implantação de várias instâncias da mesma aplicação no mesmo servidor destino, observando o perfeito funcionamento de todas as instâncias implantadas assim validando a implantação de múltiplas instâncias.

4 Resultados obtidos

Este capítulo aborda sobre os resultados encontrados, a partir da proposta de definição e preparação dos estudos, como dito no capítulo 3 as aplicações escolhidas como piloto, para terem a sua instalação automatizada foram Owncloud e Wordpress como visto em na seção , seguindo a escolha da ferramenta Shak como arquitetura inicial. Todo o trabalho foi organizado na ferramenta Gitlab no projeto Shak disponível no endereço nele contém todo o código fonte disponível.

4.1 Wordpress

De acordo com a documentação oficial (WORDPRESS, 2015) Wordpress é uma plataforma semântica de vanguarda para publicação pessoal, com foco na estética, nos Padrões web e na usabilidade e ao mesmo tempo é um software livre. Wordpress é um dos maiores software de publicação de conteúdo, sendo hoje a maior plataforma de Gerenciamento de Conteúdo do mundo, com quase 70% do mercado. O wordpress foi a primeira ferramenta escolhida para automatizar a instalação, partindo dessa decisão deu início a construção da solução.

Primeiramente foram seguido os procedimentos para construção da solução definidos na seção 3.2, é necessário definir as fases e procedimentos para a implantação automatizada, seguindo as fases que compõem o processo de implantação de aplicações e de acordo com (OMG, 2006), e isso será o que o Shak automatizará, cada fase desse processo será tratada como uma subseção. No wordpress as fases e procedimentos são:

- **Planejamento**
- **Preparação e Instalação de Pacotes**
- **Configuração**
- **Configuração de múltiplas instâncias**
- **Inicialização**

4.1.1 Planejamento

O planejamento da implantação é uma fase para identificar os componentes necessários na implantação da aplicação. Definir quais são as dependências mínimas para o funcionamento de uma aplicação, tais como: banco de dados, pacotes pré-instalados e aplicações pré-configuradas. Para o wordpress foram escolhidas:

- **Pacote wordpress para o Debian:** Pacote Debian com o wordpress.
- **Pacote Nginx para o Debian:** Pacote Debian com o Nginx.
- **Pacote mysql para o Debian:** Pacote Debian com o banco de dados mysql que será usado pelo wordpress.
- **Arquivos de configuração:** Criação dos arquivos de configuração necessários para configurar o wordpress, o servidor web Nginx e o banco de dados mysql.

A instalação e execução desses recursos serão feitas nas fases seguintes.

4.1.2 Preparação e Instalação de Pacotes

São os procedimentos necessários para preparar o ambiente alvo para que o wordpress possa ser executado, isso envolve configuração do sistema operacional, instalação e configuração de dependências necessárias, e a transferência do componente para o servidor onde ele será executado.

Para o funcionamento correto do wordpress é necessário a instalação do php5, além disso também é necessário que algum servidor web, na arquitetura do Shak o servidor web padrão é o Nginx, e o banco de dados mysql.

Para solução desse procedimento bastou apenas instalar os pacotes php5-fpm, o pacote do banco de dados mysql e o pacote nginx, além disso habilitar o serviço do php5-fpm, todos esses passos são feitos antes da instalação do wordpress, pois por default se ele não encontrar essas dependências ele instala o servidor apache2.

Para executar essas instalações é preciso criar uma receita no livro de receitas do wordpress, primeiramente é necessário criar o próprio livro de receitas no Shak, para isso existe o comando rake cookbook que cria a estrutura básica de um livro de receitas que deve ser usado pelo Chef, como por exemplo:

Código 4.1 – Exemplo de criação de estrutura básica de livro de receitas do wordpress com shak

```
rake cookbook
Cookbook name: wordpress
knife cookbook create wordpress -o cookbooks/
** Creating cookbook foo in /home/thiago/Shak/cookbooks
** Creating README for cookbook: wordpress
** Creating CHANGELOG for cookbook: wordpress
** Creating metadata for cookbook: wordpress
```

Assim com a estrutura inicial, é possível declarar as dependências do wordpress dentro do arquivo default.rb dentro da pasta recipes. Nesse arquivo é aonde será declarado os pacotes que devem ser instalados, os arquivos de configuração, os serviços que precisar executar, as pastas, permissões de usuários, etc. Um exemplo de como declarar pacotes e serviços dentro de uma receita Chef é:

Código 4.2 – Exemplo de criação de serviço do mysql com o chef

```
package "mysql-server"
service "mysql" do
  action :start
end
```

4.1.3 Configuração

Como levantado no planejamento é necessário a edição de arquivos de configuração do wordpress, arquivos de configuração do banco de dados e arquivo de configuração do nginx.

O primeiro arquivo de configuração é o config.php, é nesse arquivo de configuração onde ficam as informações de banco de dados como nome do banco de dados, login do usuário do banco de dados, senha, o host do banco de dados e o diretório onde irá ficar os temas, galeria de mídias e plugins de cada instância do wordpress. Por padrão ele deve ficar no diretório /etc/wordpress/ e seu nome deve conter a seguinte estrutura: config-nomehost.php, onde o nomehost deve ser o hostname desejado. O segundo arquivo é o database.sql que é um pequeno script *SQL* que cria o banco de dados do Wordpress e dá os privilégios ao usuário desejado. Por fim o arquivo de configuração do Nginx configuração do servidor web.

Esses arquivos de configuração serão criados dentro da pasta template, com o conteúdo desejado dos arquivos de configuração, seguindo a mesma estrutura inicial que foi criada em 4.1.3, e a ação de criação é declarada dentro do arquivo default.rb, de forma semelhante como foi feito com a declaração dos pacotes. Um exemplo de como gerenciar templates numa receita Chef é:

Código 4.3 – Exemplo de criação de templates com o chef

```
template "text\_file.txt" do
source "text\_file.txt"
mode "0755"
owner "root"
group "root"
```

4.1.4 Múltiplas Instâncias

O uso de múltiplas instâncias do wordpress pode ser feito de duas maneiras, e servem para que o usuário possa ter várias instâncias de wordpress no mesmo servidor, de acordo com a documentação do wordpress é possível ter múltiplos sites e múltiplas instâncias, no caso do Shak é desejado que seja múltiplas instâncias, a partir da repetição da instalação. O wordpress recomenda que se for necessário o uso de múltiplas instâncias é necessário realmente instalar cada Wordpress, um por vez.

A arquitetura para isso não é muito complicada, porém algumas pastas do sistema precisam ser divididas, ou seja, uma para cada aplicação, essa pasta é a pasta wp-content que é a pasta aonde fica os uploads, os temas, os plugins e etc, portanto para não existir conflito entre as instâncias essas pastas devem estar devidamente separadas com seus caminhos referenciados em cada arquivo config.php, o arquivo config.php também deve ser único para cada instância. Além disso também é preciso que cada instância tenha seu banco de dados, para evitar novamente os conflitos. Para solucionar esse problema o Shak possui um recurso interessante onde cada aplicação possui um atributo id, que é um atributo único para cada instância executada pelo Shak, com isso é possível criar pastas personalizadas com o id da aplicação e também bancos de dados específicos e arquivos config.php específicos.

Por fim basta que seja criado um arquivo de configuração Nginx para cada aplicação, como visto no capitulo 2 todo arquivo de configuração do Nginx possui um bloco server, e cada bloco server equivale a um virtual hosting, por isso as aplicações serão também independentes a nível de servidor web sendo assim possível o acesso entre elas separadamente.

4.1.5 Inicialização

Após a construção da receita do wordpress é necessário testar a receita construída, para isso o Shak precisa de duas informações importantes, a primeira é a aplicação que será instalada, e segunda o hostname destino. Como é um ambiente de desenvolvimento não é preciso configurar um ip ou configurar um *DNS*, é preciso apenas adicionar o hostname desejado no arquivo /etc/hosts. Assim mesmo que esteja na sua máquina local é possível acessar um endereço mais familiar em seu navegador. Um exemplo da execução da instalação via Shak é:

Código 4.4 – Exemplo de exexução de instalação do wordpress com shak

```
Shak install wordpress hostname=wordpress.dev
```

E assim o Chef inicia o processo de instalação dos pacotes, criação dos arquivos de configuração, incia os serviços desejados, e ao fim do procedimento a aplicação já estará pronta para uso no hostname escolhido.

4.2 Owncloud

De acordo com a documentação oficial (OWNCLOUD, 2015) Owncloud é uma ferramenta para compartilhamento de arquivos, é um software livre em que é possível compartilhar um ou mais arquivos e pastas do seu computador na nuvem, e sincronizá-los com o seu servidor Owncloud, esses arquivos são imediatamente sincronizados com o servidor e disponibilizados para outros dispositivos que utilizam o ambiente de trabalho Owncloud ou app Android ou app IOS.

Também foram seguidos os mesmos passos feitos em 4.1, primeiramente foram seguidos os procedimentos para construção da solução definidos em 3.2, definindo as fases e procedimentos para a implantação automatizada, seguindo as fases que compõem o processo de implantação de aplicações e de acordo com (OMG, 2006), cada fase desse processo também será tratado como uma subseção. No Owncloud as fases e procedimentos são:

- **Planejamento**
- **Preparação e Instalação de Pacotes**
- **Configuração**
- **Configuração de múltiplas instâncias**
- **Inicialização**

4.2.1 Planejamento

Seguindo a mesma linha do wordpress, é necessário definir quais são as dependências mínimas para o funcionamento do Owncloud tais como: banco de dados, pacotes pré-instalados e aplicações pré-configuradas. Para o Owncloud foram escolhidas:

- **Pacote Owncloud para o Debian:** Pacote Debian com o Owncloud.
- **Pacote Nginx para o Debian:** Pacote Debian com o nginx.
- **Pacote Postgresql para o Debian:** Pacote Debian com o banco de dados postgresql que será usado pelo Owncloud.
- **Arquivos de configuração:** Criação dos arquivos de configuração necessários para configurar o Owncloud, o servidor web Nginx e o banco de dados postgresql.

A diferença da escolha do banco de dados em relação ao wordpress é poder aumentar o suporte a diferentes bancos de dados, na documentação do wordpress é recomendado

o uso do banco de dados mysql, porém no Owncloud fica a escolha do desenvolvedor. A escolha do postgresql vêm com a possibilidade de trabalhar com dois bancos de dados diferentes, aumentando o suporte do Shak, assim quando existirem aplicações que suportam apenas o postgresql o Shak já terá uma estrutura básica pronta. A instalação e execução desses recursos serão feitas nas fases seguintes.

4.2.2 Preparação e Instalação de Pacotes

Para o funcionamento correto do Owncloud é necessário a instalação do php5, além disso também é necessário que algum servidor web, na arquitetura do Shak o servidor web padrão é o Nginx, e o banco de dados Postgresql.

Para solução desse procedimento foi necessário instalar os pacotes php5-fpm, o banco de dados postgresql e o pacote nginx, além disso habilitar o serviço do php5-fpm, também foi necessário instalar o pacote php5-pgsql que é um módulo para conexões de banco de dados postgresql, necessário para o funcionamento de aplicações php com o postgresql.

Para executar essas instalações é preciso criar uma receita no livro de receitas do owncloud, bastando executar novamente o comando rake cookbook que cria a estrutura básica de um livro de receitas que deve ser usado pelo Chef, como por exemplo:

Código 4.5 – Exemplo de criação de estrutura básica de livro de receitas do owncloud com shak

```
rake cookbook
Cookbook name: owncloud
knife cookbook create owncloud -o cookbooks/
** Creating cookbook foo in /home/thiago/Shak/cookbooks
** Creating README for cookbook: owncloud
** Creating CHANGELOG for cookbook: owncloud
** Creating metadata for cookbook: owncloud
```

Assim com a estrutura inicial, é possível declarar as dependências do owncloud dentro do arquivo default.rb dentro da pasta recipes. Nesse arquivo é aonde será declarado os pacotes que devem ser instalados, os arquivos de configuração, os serviços que precisar executar, as pastas, permissões de usuários, etc. Um exemplo de como declarar pacotes e serviços dentro de uma receita Chef é:

Código 4.6 – Exemplo de como habilitar serviço do postgresql com chef

```
package "postgresql"
service "postgresql" do
  action :start
```

end

4.2.3 Configuração

Como levantado no planejamento é necessário a edição de arquivos de configuração do owncloud, arquivos de configuração do banco de dados e arquivo de configuração do Nginx.

O primeiro arquivo de configuração é o autoconfig.php, é nesse arquivo de configuração onde ficam as informações de banco de dados como nome do banco de dados, login do usuário do banco de dados, senha, o host do banco de dados e o diretório onde irá ficar os arquivos de configuração do owncloud, além disso o login e senha do administrador, isso é necessário apenas para o primeiro acesso, ou seja, o usuário fará o login automaticamente quando o owncloud terminar a instalação, após isso a recomendação é trocar a senha do administrador.

Além disso é necessário criar a pasta de conteúdos públicos do owncloud, que por padrão devem ficar em /etc/owncloud. O segundo arquivo é o postgresql-conf.sql que é um pequeno script *SQL* que cria o banco de dados do owncloud e dá os privilégios ao usuário desejado. Por fim o arquivo de configuração do Nginx configuração do servidor web.

4.2.4 Múltiplas Instâncias

Diferentemente do wordpress o owncloud não suporta múltiplas instâncias, porém isso não foi um impeditivo na execução do trabalho, com uma busca encontrou-se uma discussão no repositório oficial do owncloud relacionado a implementação dessa funcionalidade, a discussão está disponibilizada em , nela existia uma proposta de solução ao fato de não existir a opção de múltiplas instâncias. O resultado desta discussão foi que os desenvolvedores do owncloud não acharam relevante a funcionalidade, mas que caso o desenvolvedor ache necessário ele poderia fazer essa alteração diretamente no código fonte.

Porém manter isso no Shak seria enviável, uma solução a esse problema seria enviar uma contribuição ao pacote do owncloud no Debian, onde assim a contribuição feita poderia ser facilmente utilizada por mais pessoas que queiram essa funcionalidade, bastando utilizar a versão disponibilizada nos servidores do Debian. Com isso, foi feito a contribuição que adicionaria a funcionalidade de múltiplas instâncias para o owncloud e enviado ao mantenedor do pacote do owncloud no Debian, a discussão está disponível em , a contribuição foi bem vista pelo mantenedor do pacote, e será incorporada na nova versão do owncloud no Debian.

Em paralelo ao processo da contribuição estar disponível no novo pacote Debian, foi necessário utilizar apenas instâncias únicas do owncloud, porém também foi adicionado na receita do owncloud o suporte a múltiplas instâncias, portanto assim que o owncloud suportar múltiplas instâncias via pacote Debian, será possível também criar múltiplas instâncias do owncloud.

A solução feita na receita do owncloud foi baseada na contribuição enviada, portanto os testes foram feitos gerando um novo pacote Debian do owncloud porém incorporando a contribuição feita. Os procedimentos feitos para suportar múltiplas instâncias no owncloud são bem semelhantes aos do wordpress, criando um diretório de dados para cada aplicação, um arquivo de configuração para cada aplicação e um banco de dados para cada aplicação. Seguindo também a mesma abordagem do wordpress também será criado um virtual hosting para cada instância do owncloud.

4.2.5 Inicialização

Após a construção da receita do owncloud é necessário testar a receita construída, para isso o Shak precisa de duas informações importantes, a primeira é a aplicação que será instalada, e segunda o hostname destino, para executar a instalação do owncloud basta:

Código 4.7 – Exemplo de execução de instalação do owncloud com shak

```
shak install owncloud hostname=owncloud.dev
```

E assim o Chef inicia o processo de instalação dos pacotes, criação dos arquivos de configuração, incia os serviços desejados, e ao fim do procedimento a aplicação já estará pronta para uso no hostname escolhido.

4.3 Servidor de e-mail

Todas as outras aplicações anteriores utilizam o protocolo *HTTP* na camada de aplicação, agora com o servidor de e-mail será trabalhado um protocolo diferente, como visto no capítulo 2 de acordo (KUROSE et al., 2010) a camada de aplicação é aonde residem as aplicações de rede e seus protocolos, dentre eles o protocolo *SMTP* que provê mensagens de correio eletrônico.

Servidores de e-mail formam a infraestrutura do e-mail, sendo o *SMTP* o protocolo mais importante pois é o responsável por transferir as mensagens de servidores de e-mail remetentes para servidores de e-mail destinatários, ou seja, transferindo arquivos (as mensagens de e-mail) de um servidor para o outro. Porém o protocolo *SMTP* é um protocolo do tipo de envio de informações, diferente do protocolo *HTTP* que é um protocolo que recupera informações, o que acarretaria um problema em que o usuário não

conseguiria recuperar suas informações como e-mails recebidos, já que o *SMTP* apenas envia informações.

Para solucionar esse problema existem os protocolos de acesso a servidores de e-mail, que recuperam as informações do servidor de e-mail até ao usuário final. Como visto no capítulo 2 existem protocolos de acesso ao correio como *POP3*, *IMAP* ou usar até mesmo o protocolo *HTTP* servindo de protocolo para recuperar informações.

Dando continuidade na construção da solução, tanto owncloud como wordpress possuem funcionalidades que dependem de serviços de e-mail funcionando, portanto é necessário uma configuração mínima de e-mail para que funcione tais funcionalidades. Portanto a construção de um servidor de e-mail possui bastante relevância pois assim qualquer aplicação que precisar de um servidor de e-mail já poderá utilizar o servidor de e-mail que o Shak fornece.

Para a construção do servidor de e-mail também foram seguidos os mesmos passos feitos em 4.1 e 4.4, primeiramente foram seguidos os procedimentos para construção da solução definidos em 3.2, definindo as fases e procedimentos para a implantação automatizada, seguindo as fases que compõem o processo de implantação de aplicações e de acordo com (OMG, 2006), cada fase desse processo também será tratado como uma subseção. No servidor de e-mail as fases e procedimentos são:

- **Planejamento**
- **Preparação e Instalação de Pacotes**
- **Configuração**
- **Inicialização**

4.3.1 Planejamento

Seguindo o mesmo raciocínio das aplicações anteriores, é necessário definir quais são as dependências mínimas para o funcionamento do servidor de e-mail, como vimos é necessário configurar um servidor que utiliza o procolo *IMAP* ou *POP3*, também é necessário a configuração de um agente de transferência de e-mails e também alguma ferramenta para controle de spam.

Primeiramente ficou definido que o protocolo escolhido seria o protocolo *IMAP*, já que o protocolo *IMAP* é um protocolo online, ou seja, ele se conecta ao servidor e realiza o download das mensagens, depois ainda mantém a conexão para que as alterações e mensagens novas sejam atualizadas em tempo real, diferentemente do protocolo *POP3* que é um protocolo offline, onde após o download das mensagens encerra a conexão.

Outra vantagem do *IMAP* em relação ao *POP3* é que o *IMAP* mantém uma cópia de mensagens no servidor, ideal para quem precisa acessar os e-mails de mais de um local.

O servidor *IMAP* escolhido foi o Dovecot que é um servidor de e-mail *IMAP*, além disso dovecot é um software livre simples de configurar, requer nenhuma administração especial e ele usa muito pouca memória (DOVECOT, 2015). O agente de transferência de e-mails escolhido foi o Postfix, que é um software livre para envio e entrega de e-mails. Sua escolha foi feita pela facilidade de integração com o Dovecot, Postfix é quem cuidará do método de entrega de e-mail utilizando *SMTP* como protocolo de transferência de e-mails. Por fim a aplicação que servirá como anti-spam será o Apache SpamAssassin que é uma plataforma anti-spam que que dá aos administradores de servidor de e-mail, um filtro para classificar e-mails e bloquear os e-mails que julgarem como spam (SPAMASSASIN, 2015).

- **Pacote dovecot para o Debian:** Pacote Debian com o dovecot.
- **Pacote postfix para o Debian:** Pacote Debian com o postfix.
- **Pacote spamassasin para o Debian:** Pacote Debian com o spamassasin.
- **Arquivos de configuração:** Criação dos arquivos de configuração necessários para configurar o spamassasin, dovecot e postfix.

4.3.2 Preparação e Instalação de Pacotes

Para o funcionamento correto do servidor de e-mail são necessários vários pacotes e com isso é possível separar as dependências dos pacotes necessários para cada ferramenta escolhida para compor o servidor de e-mail. Primeiramente para o postfix são necessários os pacotes postfix e bsd-mailx, além de habilitar o serviço do postfix, para o dovecot é necessário instalar o pacote dovecot-imadp além de habilitar o serviço do dovecot, para o spamassassin são necessários os pacotes spamassassin e o pacote spamc e ativar o serviço do spamassassin.

4.3.3 Configuração

Como levantado no planejamento é necessário a edição de arquivos de configuração de todas as ferramentas, existem arquivos de configuração para cada uma delas, inclusive o dovecot que precisa saber que estamos utilizando o postfix como agência de transferência de e-mails.

Primeiramente, para essa configuração específica do dovecot, foi utilizado 5 arquivos de configuração, o primeiro é o 10-mail.conf que é um arquivo de configuração para indicar onde será o local no qual os e-mails ficarão. O segundo arquivo é o 10-ssl.conf que é

o arquivo em que possui os caminhos do certificado *SSL* e da chave ssl, sendo o certificado pode ser lido por todos e o arquivo de chave apenas para um usuário root ou que esteja no grupo de usuários ssl-cert. O terceiro arquivo é o arquivo 11-postfix-auth.conf, arquivo responsável por indicar o caminho do arquivo de autenticação do postfix. O quarto arquivo é o arquivo 20-imap.conf onde será indicado o tamanho máximo de conexões por ip permitidas no servidor. Por fim o arquivo 20-disable-imap-non-ssl.conf que permitirá que o imap utilize sempre uma conexão segura utilizando o IMAPS (IMAP + *SSL*), assim bloqueando qualquer conexão ou tentativa na porta imap comum.

Para o posfix são apenas dois arquivos de configuração, o main.cf e o master.cf, o main.cf é aonde se configura os parâmetros mínimos de configuração, lá é aonde fica as configurações que filtram os e-mails com spamassassin e algumas configurações do *SMTP*, no master.cf definimos como um programa cliente se conecta a um serviço, e qual o programa que é executado quando um serviço é solicitado. Neste caso habilitamos recursos como: smtpd usar TLS e autenticação, caminho dos arquivos de chave e certificados o destino das mensagens como o host do servidor de e-mail, o smtpd utilizar o dovedcot e rejeitar conexões não autenticadas.

Por fim a configuração do spamassasin, são necessários dois arquivos de configuração o spamassassin que é o arquivo de configurações padrão para o spamassasin e o local.cf que são as configurações locais. No spamassassin você habilita o spamassasin e configura o caminho do arquivo de log, opções e habilitar a cronjob que será executada de tempos em tempos. Já o local.cf possui um parâmetro importante, o parâmetro required_score é um nível de 0 a 10 em que são classificados os e-mails como spam, por default esse valor é 5 porém se o usuário quiser aumentar o filtro pode aumentar para valores como 6 ou 7 até chegar em um nível de confiança em que o spamassasin cuide dos casos falsos-positivos, ou seja, e-mails que não são spam porém foram interpretados como spam.

4.3.4 Inicialização

Após a construção da receita do servidor de e-mail é necessário testar a receita construída, para isso o Shak precisa de duas informações importantes, a primeira é a aplicação que será instalada, e segunda o hostname destino, para executar a instalação do servidor de e-mail basta:

Código 4.8 – Exemplo de execução de instalação do servidor de e-mailcom shak

```
Shak install e-mail hostname=owncloud.dev
```

E assim o Chef inicia o processo de instalação dos pacotes, criação dos arquivos de configuração, incia os serviços desejados, e ao fim do procedimento a aplicação já estará pronta para uso no hostname escolhido.

Para testar o funcionamento foram feitos dois procedimentos, o primeiro era utilizar um cliente de e-mail para testes, o cliente de e-mail escolhido foi o mutt, utilizando suas funções básicas de enviar e receber e-mails. O segundo passo foi realizar conexões com o servidor de e-mail via telnet, com os seguintes comandos:

Código 4.9 – Exemplo de teste de conexão telnet no servidor imap

```
telnet localhost imap
telnet localhost imaps
```

Pela configuração do servidor de e-mail não será possível abrir uma conexão telnet em imap, porém em imaps será possível, isso serve para testar que o servidor de e-mail está barrando as conexões que não utilizem os protocolos criptografados.

4.4 MoinMoin

De acordo com a documentação oficial (OWNCLOUD, 2016) MoinMoin é uma ferramenta de construção de wikis, com uma grande comunidade de usuários, podendo criar páginas web facilmente editáveis. MoinMoin é software livre, alguns exemplos de wiki que utilizam o MoinMoin são: Wiki do debian, Wiki do python e Wiki do apache, contendo várias documentações relacionado aos seus softwares.

Para a implantação automatizada do MoinMon, foram seguidos os mesmos passos feitos em 4.1, primeiramente foram seguidos os procedimentos para construção da solução definidos em 3.2, definindo as fases e procedimentos para a implantação automatizada, seguindo as fases que compõem o processo de implantação de aplicações e de acordo com (OMG, 2006), cada fase desse processo também será tratado como uma subseção. No MoinMoin as fases e procedimentos são:

- **Planejamento**
- **Preparação e Instalação de Pacotes**
- **Configuração**
- **Configuração de múltiplas instâncias**
- **Inicialização**

4.4.1 Planejamento

Primeiramente é necessário definir quais são as dependências mínimas para o funcionamento do MoinMoin tais como: banco de dados, pacotes pré-instalados e aplicações pré-configuradas. Para o MoinMoin são elas:

- **Pacote python-moinmoin para o Debian:** Pacote Debian com o MoinMoin.
- **Pacote nginx para o Debian:** Pacote Debian com o nginx.
- **Pacote uwsgi para o Debian:** Pacote Debian com o servidor de aplicação web uwsgi, com suporte para aplicações escritas em python, utilizando o plugin uwsgi-plugin-python, também disponível como um pacote debian.
- **Arquivos de configuração:** Criação dos arquivos de configuração necessários para configurar o Uwsgi, o servidor web Nginx e o servidor web uwsgi.

Diferentemente das outras aplicações web, moinmoin não utiliza banco de dados, A diferença da escolha do banco de dados,

4.4.2 Preparação e Instalação de Pacotes

Para o funcionamento correto do moinmoin é necessário a instalação do python, além disso também é necessário que algum servidor web, no caso, utilizaremos o nginx junto ao uwsgi, também é necessário a configuração do moinmoin a partir de seus arquivos de configuração, mywiki.py e farmconfig.py.

Para solução desse procedimento foi necessário instalar os pacotes uwsgi, uwsgi-plugin-python, e o pacote do moinmoin, chamado python-moinmoin Para executar essas instalações é preciso criar uma receita no livro de receitas do moinmoin, bastando executar novamente o comando rake cookbook que cria a estrutura básica de um livro de receitas que deve ser usado pelo Chef, como por exemplo:

Código 4.10 – Exemplo de criação de estrutura básica de livro de receitas do moinmoin com shak

```
rake cookbook
Cookbook name: moinmoin
knife cookbook create moinmoin -o cookbooks/
** Creating cookbook foo in /home/thiago/Shak/cookbooks
** Creating README for cookbook: moinmoin
** Creating CHANGELOG for cookbook: moinmoin
** Creating metadata for cookbook: moinmoin
```

Assim com a estrutura inicial, é possível declarar as dependências do moinmoin dentro do arquivo default.rb dentro da pasta recipes. Nesse arquivo é aonde será declarado os pacotes que devem ser instalados, os arquivos de configuração, os serviços que precisar executar, as pastas, permissões de usuários, etc.

4.4.3 Configuração

Como levantado no planejamento é necessário a edição de arquivos de configuração do moinmoin, arquivos de configuração do uwsgi e arquivo de configuração do Nginx.

O primeiro arquivo de configuração é o moin.wsgi, arquivo onde é necessário informar, por exemplo, o caminho dos arquivos de configuração do moinmoin, no caso de um pacote debian, esse caminho por padrão é /etc/moin. Ainda relacionado ao uwsgi, também é necessário editar o arquivo uwsgi.ini, que é o arquivo de configuração do uwsgi para sua aplicação, nele é possível adicionar informações importantes, como por exemplo aonde será o arquivo de log do uwsgi, aonde será criado seu socket, e a quantidade máxima de requisições.

Agora os arquivos de configuração da wiki, o primeiro o mywiki.py, onde é necessário informar o nome da sua wiki, seu diretório de dados, ou seja, informações básicas. Agora o arquivo farmconfig.py é o responsável pelos links da barra de navegação e responsável por configurar múltiplas instâncias.

Além disso é necessário criar a pasta de conteúdos do moinmoin, que por escolha devem ficar em /var/lib/moin, além disso para que o nginx funcione corretamente com o uwsgi é necessário indicar o socket do uwsgi, que foi indicado no uwsgi.ini.

4.4.4 Múltiplas Instâncias

MoinMoin possui nativamente a funcionalidade que permite a criação de múltiplas instâncias de wiki no mesmo servidor, que é chamado de farmconfig, no qual você adiciona em um arquivo de configuração, chamado farmconfig.py, todas as wikis com seus respectivos identificadores e domínios, e assim o MoinMoin consegue gerenciar as wikis separadamente. O grande desafio é fazer com que o Shak entenda que existem outras wikis já configuradas, já que para utilizar esta funcionalidade, é necessário alterar o arquivo de configuração, ou seja, a cada nova instância o arquivo deve ser atualizado com as novas informações de wiki, sem remover as informações das outras wikis anteriormente inseridas. Para isso, é adicionado numa lista de wikis, para que a cada vez que surja uma wiki nova apenas as informações novas nessa lista são inseridas no arquivo farmconfig.py.

4.4.5 Inicialização

Após a construção da receita do owncloud é necessário testar a receita construída, para isso o Shak precisa de duas informações importantes, a primeira é a aplicação que será instalada, e segunda o hostname destino, para executar a instalação do owncloud basta:

Código 4.11 – Exemplo de execução de instalação do owncloud com shak

```
shak install moinmoin hostname=moinmoin.dev
```

E assim o Chef inicia o processo de instalação dos pacotes, criação dos arquivos de configuração, incia os serviços desejados, e ao fim do procedimento a aplicação já estará pronta para uso no hostname escolhido.

4.5 Segurança

Na implantação das aplicações foram feitos dois procedimentos de segurança, o primeiro é forçar as aplicações web a sempre utilizarem o protocolo *HTTP* e a segunda foi forçar o servidor de e-mail a não permitir a conexão via protocolos sem criptografia, neste caso sendo o protocolo imap utilizando o protocolo imaps. Para que isso fosse possível foi necessário gerar um certificado *SSL*, certificados *SSL* são necessários para que um determinado serviço opere com suporte a conexão segura por meio de criptografia. É de conhecimento do autor que a melhor forma é obter um certificado assinado por uma certificadora registrada, porém inicialmente foi trabalhado apenas com certificados autoassinados.

Para suprir a necessidade das aplicações, cada aplicação terá um certificado para si, além disso o servidor também terá o seu certificado autoassinado, isso foi necessário pois a ideia é de as aplicações possam ser instaladas em diferentes servidores, por isso é necessário garantir que cada servidor possua o seu certificado. Para adicionar esse novo suporte ao Shak foi necessário criar uma receita Chef, para que possa gerenciar os certificados *SSL*. Optou-se por utilizar a ferramenta openssl para geração das chaves e certificados, o openssl é uma ferramenta de implementação do Transport Layer Security (TLS) e Secure Sockets Layer (*SSL*), além de ser uma biblioteca de propósito geral de criptografia (OPENSSL, 2015).

Esse novo componente no Shak é composto basicamente do pacote openssl e da geração de certificados autoassinados utilizando o openssl, são gerados certificados para cada aplicação e para o servidor, o padrão do caminho onde os certificados são gerados é etc/ssl/certs/hostname.pem onde hostname é o endereço do servidor e o caminho onde as chaves são geradas é etc/ssl/private/hostname.key.

Com as chaves geradas basta configurar as aplicações indicando os caminhos dos certificados e das chaves, para forçar as aplicações web a utilizarem o *HTTP* foi necessário fazer uma configuração específica no servidor web Nginx:

Código 4.12 – Exemplo de arquivo de configuração do Nginx para aplicações web no shak

```
server {
    server_name <%= @hostname %>;
```

```
    rewrite      \^https://\$server\_name\$request\_uri? permanent;
}

server {
    server_name <%= @hostname %>;

    listen 443 ssl;
    ssl_certificate         /etc/ssl/certs/<%= @hostname %>.pem;
    ssl_certificate_key     /etc/ssl/private/<%= @hostname %>.key;

    access_log              /var/log/nginx/<%= @hostname %>.access.log;
    error_log               /var/log/nginx/<%= @hostname %>.error.log;

    include /var/lib/Shak/etc/nginx/<%= @hostname %>/*.conf;
}
```

Com isso todas as requisições serão forçadas a utilizar a porta 443 que é a porta TCP padrão para sites que utilizam *HTTP* e *SSL*, assim as aplicações utilizarão um protocolo mais seguro utilizando criptografia dos dados, utilizando o certificado autoassinado que foi gerado para o servidor específico.

4.6 Protótipo da aplicação web

Outro resultado obtido neste trabalho foi a construção de um protótipo para a interface web da aplicação, como um dos objetivos do Shak é possuir uma interface web para que os usuários não precisem utilizar um terminal para realizar as ações, também foi feito um protótipo funcional, para ser aplicado no shak. O protótipo foi feito na ferramenta chamada Pingendo, que é um software livre utilizado para construir protótipos funcionais. Além de ter um protótipo funcional, a ferramenta Pingendo também gera o código *HTML* e *CSS*, que pode ser aproveitado para a construção do layout do Shak. A sugestão inicial foi feita utilizando *HTML* e *CSS* utilizando bootstrap 3, que é um framework para criar aplicações responsivas na web. A versão inicial está disponível em .

5 Conclusão

A gerência de configuração de software é uma importante área da engenharia de software, visto que o software precisa estar implantado e disponível para uso, para que ofereça valor ao usuário final. Uma atividade importante neste cenário é a implantação automatizada de software, que facilita a instalação de ferramentas, principalmente para usuários que não possuam conhecimento técnico para realizar instalações complexas.

Observou-se que outros trabalhos já utilizam de conceitos de implantação automatizada, porém nenhuma delas voltados a aplicações em sistemas Debian GNU/Linux, que contém pacotes de algumas aplicações web livres bastante conhecidas, como wordpress e owncloud. Porém, foi visto que a apenas a instalação do pacote não é suficiente, existem ainda, algumas atividades a serem feitas, e nem sempre o usuário detém o conhecimento necessário para configurar as aplicações, com isso, dado o objetivo deste trabalho, e as motivações, foi feito evolução da ferramenta Shak, para que usuários possam instalar aplicações com apenas um clique num sistema Debian GNU/Linux.

No decorrer deste trabalho, alguns resultados foram alcançados. Primeiramente, o pesquisador estabeleceu o conhecimento sobre implantação automatizada de software e tecnologias que permitam tal implementação, como por exemplo o Chef, sendo possível automatizar todo o processo de implantação de um software, desde o planejamento até a execução. Também foi conhecido ferramentas que são utilizadas para implantação de aplicações de forma automatizada, como por exemplo o JuJu e o Bitnami, que ajudam usuários que não possuem conhecimento a instalarem aplicações com facilidade.

Além disso, também foram adquiridos os conhecimentos sobre *DevOps*, que é um conjunto de práticas e princípios para a implantação ágil de software, que foram bastante úteis ao trabalho, pois também trata de automação de implantação de software. Também foi adquirido conhecimentos sobre hospedagem virtual, que é uma forma de manter várias aplicações no mesmo servidor com um único endereço IP e vários domínios diferentes.

Em relação a construção de solução, foram realizadas melhorias na ferramenta Shak, provendo quatro novas aplicações, sendo elas Owncloud, Wordpress, MoinMoin e servidor de e-mail, todas sendo implantadas automaticamente e levando em consideração o uso de protocolos seguros como *HTTP* junto ao SSL e SMTP junto ao SSL. Logo também foi adicionado a ferramenta Shak suporte a criação de certificados autoassinados, permitindo que as aplicações possam usar certificados para configurar tais protocolos. Ainda foi adicionado ao Shak o suporte a as aplicações web sempre utilizarem o protocolo HTTPS, evitando assim implantações inseguras, independente da ferramenta.

Também foi configurado o suporte a múltiplas instâncias nas aplicações disponíveis

no shak, o Wordpress e MoinMoin já previa tal suporte, bastando configurá-lo, porém no Owncloud foi necessário o envio de uma correção para o mantenedor do pacote Debian, visto que é uma funcionalidade nova no Owncloud, permitindo assim a criação de múltiplas instâncias do Owncloud, em ambos os casos foi utilizado a técnica de hospedagem virtual, disponibilizada pelo servidor web Nginx. Além disso, foi proposto um protótipo da interface web para a ferramenta, que seria a forma dos usuários acessarem a ferramenta sem precisar utilizar um terminal.

Por fim, todos esses resultados são importantes para o objetivo deste trabalho, que vão desde compreender como é o processo de implantação de um software, a evolução da ferramenta Shak, e até os cuidados a se tomar ao implantar uma aplicação, sendo assim, contribuindo para o resultado deste trabalho.

5.1 Trabalhos Futuros

Algumas melhorias na ferramenta Shak não puderam ser implementadas no contexto deste trabalho, destaca-se o suporte a servidores autoassinados utilizando let's encrypt, que trariam ganhos para os usuários finais, visto que os certificados autoassinados que utilizam a ferramenta openssl não são reconhecidos automaticamente pelos navegadores, precisando ainda, de uma autorização do usuário. Já com let's encrypt esse problema não aconteceria, visto que os certificados gerados a partir desta tecnologia é reconhecido pelos navegadores mais modernos.

Outra melhoria seria o suporte a criação de máquinas virtuais de forma automatizada, visto que ainda é necessário que o usuário instale a ferramenta no servidor destino, uma solução viável é que o usuário possa instalar a partir de seu computador pessoal, apontando apenas o endereço do servidor destino, tanto a partir do terminal, como a partir da interface web, facilitando ainda mais o uso da ferramenta para seus usuários, ou até mesmo utilizar a ferramenta Shak como um serviço, sendo assim, disponibilizando uma versão do Shak na nuvem, para que qualquer usuário com conexão a internet e um servidor disponível pudesse instalar aplicações web em que o Shak dê suporte.

Referências

APACHE. *Apache Virtual Hosting Documentation*. [S.l.], 2015. Disponível em: <https://httpd.apache.org/docs/2.2/en/vhosts/>. Citado na página 25.

ARAÚJO, T. C. *AppRecommender: um recomendador de aplicativos GNU/Linux*. Tese (Doutorado) — Universidade de Sao Paulo, 2011. Citado na página 24.

BENDIX, L.; KOJO, T.; MAGNUSSON, J. Software configuration management issues with industrial opensourcing. In: *2011 IEEE Sixth International Conference on Global Software Engineering Workshop*. [S.l.: s.n.], 2011. p. 85–89. ISSN 2329-6305. Citado na página 17.

BITNAMI. *What is Bitnami*. [S.l.], 2015. Disponível em: <https://bitnami.com/learn_more>. Citado 2 vezes nas páginas 31 e 35.

CARZANIGA, A. et al. A characterization framework for software deployment technologies. Department of Computer Science,University of Colorado,Boulder,CO, 1998. Citado 2 vezes nas páginas 18 e 23.

CASTELLS, M. *A Galáxia Internet: reflexões sobre a Internet, negócios e a sociedade*. [S.l.]: Zahar, 2003. Citado na página 14.

DEBIAN. *Debian Policy Manual*. [S.l.], 2015. Disponível em: <https://www.debian.org/doc/debian-policy/>. Citado na página 24.

DOVECOT. *Dovecot, Secure IMAP server*. [S.l.], 2015. Disponível em: <http://www.dovecot.org/>. Citado na página 52.

ENTERPRISE, G. *Gitlab Documentation*. [S.l.], 2015. Disponível em: <https://about.gitlab.com/documentation/>. Citado na página 33.

FEITELSON, D.; FRACHTENBERG, E.; BECK, K. Development and deployment at facebook. *Internet Computing, IEEE*, v. 17, n. 4, p. 8–17, July 2013. ISSN 1089-7801. Citado na página 33.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de pesquisa*. [S.l.]: PLAGEDER, 2009. Citado na página 30.

GILES, D. *Psychology of the media*. [S.l.]: Palgrave Macmillan, 2010. Citado na página 14.

HTTERMANN, M. *DevOps for developers*. [S.l.]: Apress, 2012. Citado na página 21.

HUMBLE, J.; FARLEY, D. Continuous delivery: Reliable software releases through build, test, and deployment automation. In: ADDISON-WESLEY (Ed.). [S.l.: s.n.], 2010. Citado na página 20.

HUMMER, W. et al. Testing idempotence for infrastructure as code. In: _____. *Middleware 2013: ACM/IFIP/USENIX 14th International Middleware Conference, Beijing, China, December 9-13, 2013, Proceedings*. Berlin, Heidelberg: Springer Berlin

Heidelberg, 2013. p. 368–388. ISBN 978-3-642-45065-5. Disponível em: <http://dx.doi.org/10.1007/978-3-642-45065-5_19>. Citado na página 22.

IEEE. Ieee standard for configuration management in systems and software engineering - redline. *IEEE Std 828-2012 (Revision of IEEE Std 828-2005) - Redline*, p. 1–126, March 2012. Citado na página 17.

INMETRO. *Automatic Certificate Management Environment (ACME)*. [S.l.], 2015. Disponível em: <https://tools.ietf.org/html/draft-barnes-acme-04>. Citado na página 29.

JUJU. *What is juju?* [S.l.], 2015. Disponível em: <https://jujucharms.com/docs/stable/about-juju>. Citado na página 31.

KUROSE, J. F. et al. *Redes de computadores ea Internet: Uma abordagem top-down*. [S.l.]: Pearson, 2010. Citado 4 vezes nas páginas 26, 27, 28 e 50.

LEITE, L. A. F. Implantação automatizada de composições de serviços web de grande escala. In: . USP, São Paulo: [s.n.], 2014. Citado 4 vezes nas páginas 18, 23, 31 e 32.

MANTYLA, M.; VANHANEN, J. Software deployment activities and challenges - a case study of four software product companies. In: *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*. [S.l.: s.n.], 2011. p. 131–140. ISSN 1534-5351. Citado 2 vezes nas páginas 18 e 23.

NGINX. *Nginx Documentation*. [S.l.], 2015. Disponível em: <https://www.nginx.com/resources/wiki/>. Citado na página 35.

OMG. Deployment and configuration of component-based distributed applications specification. 2006. Disponível em: <http://www.omg.org/spec/DEPL/4.0/>. Citado 6 vezes nas páginas 19, 38, 43, 47, 51 e 54.

OPENSSL. *OpenSSL Documentation*. [S.l.], 2015. Disponível em: <https://www.openssl.org/>. Citado na página 57.

OWNCLOUD. *User Manual*. [S.l.], 2015. Disponível em: <https://doc.owncloud.org/server/9.0/user_manual/>. Citado na página 47.

OWNCLOUD. *User Manual*. [S.l.], 2016. Disponível em: <https://moinmo.in/>. Citado na página 54.

RAHMAN, A. et al. Synthesizing continuous deployment practices used in software development. In: *Agile Conference (AGILE), 2015*. [S.l.: s.n.], 2015. p. 1–10. Citado na página 18.

RUBYGEMS. *What is a gem ?* [S.l.], 2015. Disponível em: <http://guides.rubygems.org/what-is-a-gem/>. Citado 2 vezes nas páginas 35 e 36.

SANDSTORM.IO. *What is Sandstorm?* [S.l.], 2015. Disponível em: <https://docs.sandstorm.io/en/latest/>. Citado 2 vezes nas páginas 31 e 35.

SHARMA, T.; FRAGKOULIS, M.; SPINELLIS, D. Does your configuration code smell? In: *Proceedings of the 13th International Conference on Mining Software Repositories*. New York, NY, USA: ACM, 2016. (MSR '16), p. 189–200. ISBN 978-1-4503-4186-8. Disponível em: <http://doi.acm.org/10.1145/2901739.2901761>. Citado na página 22.

SPAMASSASIN, A. *Welcome to SpamAssasin*. [S.l.], 2015. Disponível em: <http://spamassassin.apache.org/>. Citado na página 52.

SPINELLIS, D. Don't install software by hand. *IEEE Software*, IEEE Computer Society, Los Alamitos, CA, USA, v. 29, n. 4, p. 86–87, 2012. ISSN 0740-7459. Citado 2 vezes nas páginas 14 e 22.

SPINELLIS, D. Don't install software by hand. *Software, IEEE*, v. 29, n. 4, p. 86–87, July 2012. ISSN 0740-7459. Citado 3 vezes nas páginas 15, 21 e 23.

TERCEIRO, A. Shak: uma proposta para descentralização dos serviços de internet. Artigo não publicado, 2015. Citado 2 vezes nas páginas 14 e 15.

TERRA, C. F. *Comunicação corporativa digital: o futuro das relações públicas na rede.* Tese (Doutorado) — Universidade de São Paulo, 2006. Citado na página 15.

VAZ, P. As esperanças democráticas e a evolução da internet. *Revista FAMECOS: mídia, cultura e tecnologia*, v. 1, n. 24, 2006. ISSN 1980-3729. Disponível em: <http://200.144.189.42/ojs/index.php/famecos/article/view/391/320>. Citado na página 14.

VIRMANI, M. Understanding devops bridging the gap from continuous integration to continuous delivery. In: *Innovative Computing Technology (INTECH), 2015 Fifth International Conference on*. [S.l.: s.n.], 2015. p. 78–82. Citado 2 vezes nas páginas 20 e 21.

WORDPRESS. *Documentação Wordpress*. [S.l.], 2015. Disponível em: <http://codex.wordpress.org/pt-br:P%C3%A1gina_Inicial>. Citado na página 43.