UNIVERSIDADE FEDERAL DO PAMPA

CIÊNCIA DA COMPUTAÇÃO

LUCAS MENDES RIBEIRO ARBIZA

UMA ABORDAGEM PARA O GERENCIAMENTO INTEGRADO DE MÁQUINAS VIRTUAIS E SERVIÇOS DE REDES

Trabalho de Conclusão de Curso

Alegrete

2011

LUCAS MENDES RIBEIRO ARBIZA

UMA ABORDAGEM PARA O GERENCIAMENTO INTEGRADO DE MÁQUINAS VIRTUAIS E SERVIÇOS DE REDES

Trabalho de Conclusão de Curso apresentado como parte das atividades para obtenção do título de bacharel em Ciência da Computação na Universidade Federal do Pampa.

Orientador: Prof. Me. Diego Luís Kreutz

Alegrete

2011

LUCAS MENDES RIBEIRO ARBIZA

UMA ABORDAGEM PARA O GERENCIAMENTO INTEGRADO DE MÁQUINAS VIRTUAIS E SERVIÇOS DE REDES

Trabalho de Conclusão de Curso apresentado como parte das atividades para obtenção do título de bacharel em Ciência da Computação na Universidade Federal do Pampa.

Trabalho apresentado e aprovado em 1º de julho de 2011.

Banca examinadora:

Prof. Msc. Diego Luis Kreutz
Orientador
Ciência da Computação - UNIPAMPA

Prof. Dra. Andrea Schwertner Charão
Departamento de Eletrônica e Computação - UFSM

Prof. Dra. Márcia Cristina Cera
Ciência da Computação - UNIPAMPA

RESUMO

A virtualização está sendo amplamente utilizada em empresas de todo o mundo pelos benefícios que proporciona e é apontada como uma tendência que vem mudando a forma de planejar e gerenciar a infraestrutura de TI. Conjuntos de máquinas virtuais são criados e gerenciados, de forma dinâmica e automatizada, através de soluções de virtualização com o objetivo de disponibilizar sistemas e serviços. Para auxiliar a administração de ambientes computacionais com grande número de computadores surgiram ferramentas que automatizam a instalação, configuração, manutenção e monitoramento de sistemas e serviços em larga escala. Soluções de virtualização e ferramentas de gerenciamento de sistemas e serviços são utilizadas em um mesmo conjunto de servidores, porém são utilizadas e administradas separadamente aumentando o trabalho de administradores de redes que precisam conhecer bem ferramentas distintas a fim de tirar maior proveito de sua utilização. Este trabalho propõe uma solução onde o provisionamento e o gerenciamento de servidores virtuais, com serviços instalados e configurados, são automatizados a partir de informações contidas em um repositório. O objetivo desta proposta é simplificar o gerenciamento de servidores virtuais através de uma solução que integre o gerenciamento de máquinas virtuais e o gerenciamento de serviços e possa ser utilizada sem a necessidade de especialização. Esta solução possibilita ao usuário criar novos servidores virtuais selecionando *templates* nos quais são descritos sistemas e serviços e fornecendo informações de rede e arquivos de configuração que configuram o servidor virtual para que atenda uma demanda específica. Baseando-se nas informações contidas nos *templates* selecionados e nas informações fornecidas pelo usuário o sistema automatiza o provisionamento de um novo servidor virtual configurado e pronto para uso. A solução replica atualizações realizadas em *templates* para os servidores virtuais criados a partir dos *templates* atualizados, sem a necessidade de intervenção manual.

Palavras-chave: Virtualização, gerenciamento, servidores virtuais, serviços, automação.

ABSTRACT

Virtualization has being largely used in companies around the world due to the benefits it provides and it is pointed as a trend that is changing the way the IT infrastructure is plained and managed. Sets of virtual machines are dinamically and automaticaly created and managed by virtualization solutions in order to provid systems and services. To assist the management of computing environments with a large number of computers, some tools emerged that automate installation, configuration and manteinance of systems and services in large scale. Virtualization solutions and tools for systems and services management are used to manage the same set of servers, but these solutions and tools are used and managed separately, this increases network administrators work that need to know deeply different tools to get the most out of its use. This paper proposes a solution where the provisioning and management of virtual servers with services installed and configured are automated using information contained in a repository. This proposal aims at simplify virtual servers management through a solution that integrate virtual servers and services management and may be used without the need of expertise. This solution enables the user to create new virtual servers by selecting templates where systems and services are described and providing network information and settings files that configure out the virtual server to meet a specific demand. Based on the information contained in the selected templates and on the informations provided by user the system automates the new virtual server provisioning configured and ready to use. The solution replicates updates realized in templates to the virtual servers created from the updated templates without the need of manual intervention.

Keywords: Virtualization, management, virtual servers, services, automation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Program Interface*

CEO – *Chief Executive Officer*

CLI – *Command Line Interface*

CPU – *Central Processing Unit*

DHCP – *Dynamic Host Configuration Protocol*

DRBD – *Distributed Replicated Block Device*

GUI – *Graphical User Interface*

HD – *Hard Disc*

LVM – *Logical Volume Manager*

MMV – Monitor de máquina virtual

MV – Máquina Virtual

NTP – *Network Time Protocol*

PHP – *Hypertext Preprocessor*

RAID – *Redundant Array of Independent Discs*

SO – Sistema Operacional

SQL – *Structured Query Language*

TI – Tecnologia da Informação

SUMÁRIO

1 INTRODUÇÃO

De acordo com Gartner, citado por Marshall (2010), a virtualização já está presente em 80% das empresas de todo o mundo; a virtualização é uma tendência em tecnologia da informação e vai continuar como um desafio para infraestrutura e operações mudando a forma de planejar, gerenciar e implantar infraestrutura de TI. Diversas tecnologias e soluções surgiram e vem sendo cada vez mais aprimoradas possibilitando o gerenciamento de máquinas virtuais individualmente, bem como de grandes conjuntos de máquinas virtuais de forma autônoma.

A virtualização está presente em servidores, *desktops*, dispositivos de armazenamento, aplicações e redes (VERAS, 2011, p.4; CISCO SYSTEMS, 2006). As ferramentas para virtualização transformaram-se em soluções avançadas de gerenciamento de infraestrutura virtualizada. Elas passaram de simples ferramentas de criação de máquinas virtuais para soluções com recursos de migração em tempo de execução, monitoramento, balanceamento de carga, alta disponibilidade, otimização da utilização dos recursos de *hardware*, dentre outras funcionalidades. Os recursos e benefícios proporcionados pela virtualização, como a disponibilização elástica dos recursos computacionais e a entrega destes recursos sob demanda, viabilizam uma nova forma de prover recursos de TI, a disponibilização de software, plataforma e infraestrutura como serviços (SaaS[1], PaaS[2], IaaS[3]) (EUCALYPTUS SYSTENS, 2011).

A infraestrutura de TI de uma organização é composta de servidores físicos e virtuais que estão executando serviços constantemente em ambientes normalmente heterogêneos. Por outro lado o gerenciamento destes serviços é um dos grandes desafios que tenta ser resolvido por tecnologias, ferramentas e propostas de novas abordagens, metodologias e soluções (DASTJERDI, 2010).

Virtual appliances tem sido utilizadas como forma de contornar as complexidades do processo de provisionamento de serviços (DASTJERDI, 2010) e por serem mais fáceis de administrar (KREUTZ, 2009, p. 29). *Virtual appliances* encapsulam o estado de uma máquina virtual (SO, serviços instalados, configurações), com isso os

1 Acrônico para *Software as a Service* (*Software* como serviço)

2 Acrônico para *Platform as a Service* (Plataforma como serviço)

3 Acrônico para *Infrastructure as a Service* (Infraestrutura como serviço)

serviços podem ser providos através da instanciação de máquinas virtuais (KECSKEMETI et al., 2011).

No contexto da virtualização, máquinas virtuais e serviços são executados e gerenciados num mesmo ambiente, porém de forma isolada, ou seja, o gerenciamento de máquinas virtuais é paralelo ao gerenciamento de serviços. Máquinas virtuais são gerenciadas por soluções de virtualização que facilitam e automatizam o gerenciamento, mas grande parte dos recursos oferecidos pelas soluções de virtualização não são acessíveis às organizações de pequeno e médio porte devido ao alto custo das licenças. O gerenciamento de serviços é realizado por ferramentas que instalam, configuram e mantem serviços de forma automática e em larga escala. No entanto, estas soluções tem uma aceitação média baixa em ambientes de pequeno e médio porte, em especial pelo fato de requerem especialização para a sua utilização, o que pode implicar em recursos humanos dedicados, muitas vezes inexistentes em boa parte das organizações.

Considerando o contexto apresentado, este trabalho detalha o projeto e a prototipação de uma solução que integre o gerenciamento de máquinas virtuais e serviços que possa ser utilizada sem a necessidade de especialização em redes de pequeno porte. Nesta solução máquinas virtuais e serviços são definidos em modelos de representação de dados (*templates*) armazenados em um repositório, a partir destes dados o sistema automatiza a criação e atualização de servidores virtuais.

A solução apresentada permite aos administradores criar novos servidores virtuais selecionando *templates*. Através da edição de *templates* de serviços a solução replica as atualização a todos os *templates* que utilizam o *template* editado.

O capítulo 2 apresenta as principais soluções de virtualização utilizadas por grande parte das empresas de todo o mundo, apresenta algumas soluções que utilizam os recursos da virtualização para a criação e gerenciamento de nuvens computacionais privadas, públicas e híbridas e também as *virtual appliances* como uma nova forma de prover sistemas e serviços. Ainda no capítulo 2 são apresentadas ferramentas utilizadas para o gerenciamento de sistemas e serviços de forma autônoma.

O capítulo 3 apresenta a problemática envolvendo os custos de licenciamento de soluções de virtualização e a necessidade de especialização de pessoal para a utilização

de ferramentas de gerenciamento de sistemas e serviços. Os custos e a necessidade de pessoal especializado por vezes impede que organizações de pequeno e médio porte, com suas limitações de recursos, se beneficiem das funcionalidades de soluções de virtualização e ferramentas de gerenciamento de serviços. Com base nos problemas mencionados é então apresentada uma proposta de solução que integre o gerenciamento de máquinas virtuais e serviços em um sistema que possa ser utilizado sem a necessidade de especialização.

Os detalhes de implementação, a descrição detalhada dos componentes do sistema, os recursos utilizados e os resultado deste trabalho são apresentados no capítulo 4. O capítulo 5 apresenta as considerações finais e conclusões do trabalho desenvolvido.

2 VIRTUALIZAÇÃO, *VIRTUAL APPLIANCES* E SERVIÇOS DE REDE

Alguns anos atrás, quando a virtualização começou a ser utilizada em máquinas de produção de grandes e pequenas empresas, os produtos da Vmware (http://vmware.com/) eram a opção mais óbvia e vantajosa devido ao conjunto de funcionalidades que ofereciam, sua reputação e preço (VENEZIA, 2011, p. 1). Em 2008, em pesquisa realizada pelo *Enterprise Strategy Group* (http://www.enterprisestrategygroup.com), a VMware era apontada como a principal empresa fornecedora de soluções de virtualização possuindo 70% do mercado da época (VERAS, 2011, p. 13). Em artigo na ReadWriteWeb (http://www.readwriteweb.com) com dados mais atuais Willians (2010) afirma que a VMware possui 80% do mercado de virtualização. A VMware possui ainda a incrível marca de 100% das 100 maiores empresas do mundo serem seus clientes, segundo a revista Fortune (http://money.cnn.com/magazines/fortune), e possivelmente 96% das 1000 maiores empresas do mundo (WIKINVEST).

Apesar da superioridade numérica da VMware, empresas como Citrix (http://www.citrix.com), Microsoft (http://www.microsoft.com) e Red Hat (http://www.redhat.com/) oferecem ferramentas com características e funcionalidades bastante competitivas. Contudo a VMware ainda oferece mais opções aos seus clientes em relação a hardware e sistemas operacionais homologados (VERAS, 2011, capítulo 2).

Em um teste comparativo realizado pela InfoWorld (http://www.infoworld.com/) as soluções de virtualização Citrix XenServer 5.61, Microsoft Windows Server 2008 R2 Hyper-V, Red Hat Enterprise Virtualization for Servers 2.2 e VMware vSphere 4.1 foram comparados. Uma tabela[1] que compara 25 características entre as quatro soluções testadas foi apresentada. Dentre todas as características apenas duas funcionalidades não estão presentes nas quatro soluções, o que deixa clara a semelhança em termos de recursos providos.

Uma característica que tem se tornado mais expressiva nas soluções de virtualização é a automação. Funcionalidades como balanceamento de carga, alta

1A tabela mencionada pode ser visualizada através do link: <http://rww.readwriteweb.netdna-cdn.com/cloud/assets_c/2011/04/virtshootout.features-thumb-458x979-29190.jpg>. Acesso em: Maio de 2011.

disponibilidade, *live migration* e recuperação de desastres garantem de forma automática a disponibilidade de servidores e serviços.

2.1 Funcionalidades

A seguir serão detalhadas algumas das características comuns às soluções utilizadas no teste comparativo realizado pela InfoWorld:

Alta disponibilidade: Falhas em sistemas podem torná-lo inacessível por alguns instantes, horas ou dias. A alta disponibilidade é a capacidade de um sistema se manter disponível apesar da ocorrência de incidentes. Normalmente a alta disponibilidade está associada à tolerância a falhas e à prevenção dos prejuízos ocasionados pela indisponibilidade de sistemas (VARGAS, 2000, p. 2-3). As soluções de virtualização tomam providências como reiniciar uma máquina virtual para torná-la disponível na ocorrência de alguma falha em seu funcionamento (RED HAT, 2010, p. 3).

Balanceamento de carga: Na indisponibilidade de um dos *hosts* ou se este está sobrecarregado, as soluções de virtualização gerenciam esta situação, por exemplo, reiniciando as máquinas virtuais imediatamente em outros *hosts* ou migrando uma ou mais máquinas virtuais para *hosts* menos carregados (RED HAT).

Otimização do uso de memória: Consiste do compartilhamento da memória não utilizada entre as máquinas virtuais gerando melhora de performance (CITRIX, 2010, p. 3).

Monitoramento de performance: Disponível através da interface gráfica das soluções fornece ao usuário informações em tempo real do *host* e das máquinas virtuais.

Provisionamento de máquinas virtuais: Este processo consiste da criação e configuração de máquinas virtuais. O processo de criação de uma máquina virtual é feito através de uma interface gráfica onde primeiramente o usuário seleciona as configurações de hardware virtual ou a forma como a máquina virtual poderá utilizar o hardware real e configurações do sistema operacional. Este processo pode ser acelerado e automatizado com a utilização de *templates* nos quais estão contidas

informações relativas ao hardware e ao sistema operacional da nova máquina virtual (CITRIX, 2010, p. 3; RED HAT).

Provisionamento de serviços: Imagens, chamadas *golden images* (CITRIX, 2010, p. 3), são criadas a partir de máquinas virtuais com aplicações instaladas e configuradas. Quando uma nova máquina virtual semelhante a uma máquina existente é necessária esta nova máquina virtual é criada a partir de uma *golden image* e o sistema entrega a nova máquina virtual em menor tempo e absolutamente pronta para utilização (RED HAT).

Recuperação de desastres: É a capacidade de um sistema de autorrecuperar-se e voltar ao estado/situação em que se encontrava no momento da ocorrência de alguma falha (desastre) e então continuar em seu funcionamento normal. Mais especificamente soluções de virtualização buscam rapidez, confiabilidade, redução de custos e perdas e completa automação deste processo (VMWARE, 2011). Facilitam o planejamento das ações de recuperação com fácil configuração e realizam testes para garantir confiabilidade das configurações (CITRIX, 2010, p. 3).

Conversão *virtual-to-virtual* (v2v) e *physical-to-virtual* (p2v): Conversão *virtual-to-virtual* é a conversão de uma máquina virtual cujo formato é de outra solução ou tecnologia de virtualização em uma máquina virtual no formato da solução utilizada. Conversão *physical-to-virtual* é a conversão de um sistema rodando em uma máquina real em uma máquina virtual no formato do sistema utilizado.

***Live migration*:** Recurso que permite que uma máquina virtual hospedada em um determinado host seja transferida para outro *host* sem interrupção em sua disponibilidade, ou seja, usuários que estiverem utilizando aplicações que estejam rodando na máquina virtual migrada não percebem a realização desta operação.

***Snapshot*:** É a captura do estado atual de uma máquina virtual como se fosse uma foto instantânea. O *snapshot* armazena todos os detalhes de configuração da máquina virtual e a partir desta informações outras máquinas virtuais podem ser criadas com as mesmas configurações daquele instante no qual o *snapshot* foi obtido.

Interface de gerenciamento: Através de uma interface gráfica é possível gerenciar máquinas virtuais em um ou mais *hosts*. Soluções de virtualização provêm interfaces que, além das funções relativas ao gerenciamento de máquinas virtuais,

fornecem dados sobre os a utilização dos recursos de hardware do *host* pelo próprio sistema ou pelas máquinas virtuais. A utilização deste tipo de interface possibilita a administração centralizada de centenas de máquinas virtuais (CITRIX, 2010, p. 4).

2.2 Virtualização e Computação em nuvem

A virtualização é o meio de disponibilização de recursos por demanda e de forma elástica necessário à computação em nuvem (DASMALCHI, 2009). Virtualização e computação em nuvem estão, portanto, relacionadas.

As soluções de virtualização abordadas provêm os recursos necessários para computação em nuvem e funcionalidades para o gerenciamento autônomo de *datacenters*. Além das soluções apresentadas há outras soluções desenvolvidas para o gerenciamento de máquinas virtuais em larga escala com foco em computação em nuvem que serão denominadas neste trabalho como gerenciadores de nuvens. Os gerenciadores de nuvens como Ganeti (http://code.google.com/p/ganeti/), Eucalyptus (http://open.eucalyptus.com/), OpenNebula (http://opennebula.org/) e OpenStack (http://www.openstack.org/), assim como as soluções de virtualização, fornecem recursos para o gerenciamento de todo o ciclo de vida de um grande conjunto de máquinas virtuais, *storages* e redes virtuais de forma centralizada, porém não oferecem um conjunto de funcionalidades tão completo quanto as soluções de virtualização e utilizam, como tecnologia de virtualização para a execução de máquinas virtuais, as mesmas tecnologias (*hypervisors*) das soluções de virtualização.

O Ganeti é um projeto da Google (http://www.google.com/) para o gerenciamento de servidores virtuais em *clusters*. Suporta os *hypervisors* Xen e KVM e tecnologias como DRBD, LVM e RAID-1. Segundo a equipe de desenvolvimento a utilização desta solução resultou em um aumento drástico na eficiência de utilização de hardware, além da economia com espaço, energia e refrigeração (LINUX MAGAZINE, 2007).

O Eucalyptus surgiu como um projeto universitário no Departamento de Ciência da Computação da Universidade de Santa Bárbara (Califórnia, EUA) (http://www.cs.ucsb.edu/). Utiliza como *hypervisor* o KVM. O Eucalyptus permite a

criação de “nuvens” sem a necessidade de reequipar ou adicionar hardware especializado à estrutura de TI existente na organização (EUCALYPTUS SYSTENS, 2011).

Em 2005, também como um projeto de pesquisa, surgiu o OpenNebula que teve sua primeira versão em março de 2008. O OpenNebula “é fruto de anos de pesquisa e desenvolvimento para o gerenciamento de máquinas virtuais em larga escala em sistemas distribuídos” (OPENNEBULA PROJECT LEADS, 2011, tradução nossa).

Criado em uma parceria entre Rackspace Hosting (http://www.rackspace.com/) e NASA (*National and Aeronautics and Space Administration*, http://www.nasa.gov/), o OpenStack é uma plataforma de computação em nuvem constituída de três módulos: 1) *Compute*: para a criação de uma plataforma de computação em nuvem para o provisionamento e gerenciamento de máquinas virtuais em larga escala, 2) *Object storage*: para a criação de armazenamento de objetos redundantes e escalonáveis em *clusters* e 3) *Image Service*: para o gerenciamento de discos virtuais em sistemas distribuídos através dos quais instâncias de máquinas virtuais são criadas (OPENSTACK.ORG).

2.3 Virtual appliances

Virtual appliances são soluções pré-configuradas e prontas para uso o que proporciona economia de tempo, dinheiro, serviço e recursos gastos com o processo tradicional de instalação e manutenção de *software*. As *virtual appliances*, ao invés de instaladas em um sistema operacional, são apenas executadas em uma plataforma de virtualização como qualquer máquina virtual (VMWARE, 2010).

Virtual appliances são uma forma diferente de prover máquinas virtuais e serviços. Em seu vídeo demonstrativo a VMware afirma que as *virtual appliances* estão “mudando a forma como *software* é desenvolvido, distribuído, implantado e gerenciado” (VMWARE, 10 seg., tradução nossa).

A forma comum de implantação de serviços é custosa em termos de tempo e recursos humanos investidos, a utilização de *virtual appliances* elimina grande parte deste custo. Ao invés da criação e implantação dos serviços serem realizados localmente pela solução de virtualização as *virtual appliances* podem ser obtidas

facilmente de repositório abertos como VMware *Virtual Appliance Marketplace* (http://www.vmware.com/appliances/directory/) ou de repositórios locais como mostrado em KREUTZ (2009) e executadas localmente como mostrado na figura 1.

Há usuários e corporações trabalhando em soluções para automatizar a criação e disponibilização de *virtual appliances*, bem como tratar restrições de arquitetura de monitores de máquinas virtuais (KREUTZ, 2009).

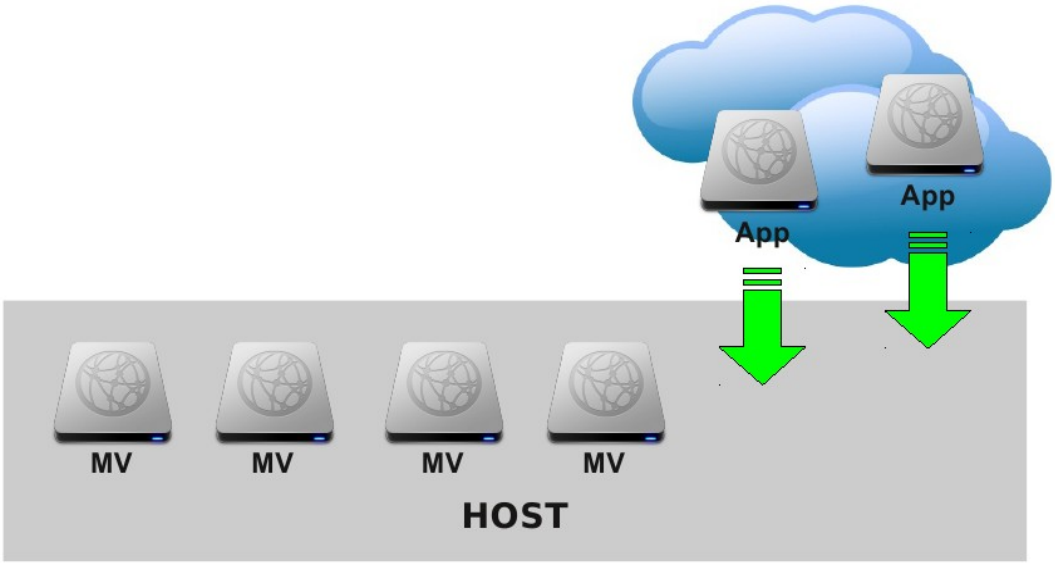


Figura 1 - Utilização de virtual appliances

2.4 Gerenciamento de serviços

O isolamento entre máquinas virtuais, a otimização da utilização dos recursos de hardware, alta disponibilidade, balanceamento de carga, recuperação de falhas são alguns benefícios providos pela virtualização que tornam bastante conveniente e comum a prática de se manter um serviço por servidor virtual. Desta forma o comprometimento de um serviço ou servidor não compromete os demais. Facilidade dificilmente aplicável sem a virtualização devido ao alto custo de hardware. Porém serviços não se limitam ao hardware, uma vez que um único serviço pode envolver diversos servidores trabalhando em conjunto.

O gerenciamento de serviços, como afirma Limoncelli, não é apenas colocar hardware e software juntos, envolve também tornar o serviço confiável, dimensionar seu crescimento, monitorá-lo, mantê-lo e suportá-lo (2007, p. 95). Para atender estes requisitos ferramentas como Cfengine (http://cfengine.com/), Puppet

(http://www.puppetlabs.com/) e Chef (http://wiki.opscode.com/display/chef/Home) auxiliam e automatizam o gerenciamento de configurações de sistemas e serviços.

O Cfengine realiza o gerenciamento de configuração de sistemas baseado em arquivos de configuração centralizados em um servidor, os servidores são definidos em classes para as quais são definidas as configuração. Com o Cfengine é possível realizar alterações em diversos servidores facilmente e a cada novo servidor adicionado à rede as configurações dos demais podem ser rapidamente aplicadas (BAUER, 2004). Quando executado esta solução realiza verificação de acordo com um período de tempo definido, verifica em que classe o servidor se encontra e aplica uma séria de ações previamente definidas (BORWICK, 2006). Com o Cfengine obtêm-se melhorias em termos de segurança e agilidade pois as atualizações são replicadas automaticamente a partir do servidor central, permissões de arquivos e seus conteúdos são conferidos, serviços são habilitados e desabilitados para que sistemas e serviços permaneçam como definidos em suas respectivas classes (BAUER, 2004).

O Puppet é um *framework* que, assim como o Cfengine, automatiza tarefas repetitivas e o processo de configuração de sistemas e serviços. Possui uma linguagem própria que possibilita a criação de módulos e manifestos. Módulos são “unidades de trechos de código Puppet reutilizáveis e compartilháveis que automatizam tarefas como configurar um banco de dados, um servidor web ou um servidor de emails.” (PUPPET, 2010, tradução nossa); manifestos são “gráficos de recursos que representam o estado desejado das máquinas” (MOERTEL, 2007, tradução nossa). Cada máquina compara seu estado atual com o estado descrito nos módulos e nos manifestos centralizados em um servidor e então o Puppet realiza as ações necessárias para que o sistema seja atualizado. O administrador não precisa se preocupar com detalhes como os comandos de atualização próprios de distribuições Debian-like, Fedora, OpenSuse, dentre outros, apenas declara uma atualização em um manifesto e o Puppet se encarrega dos detalhes de cada sistema operacional (BROCKMEIER, 2010).

Assim como o Puppet, o Chef possibilita a administração de sistemas e serviços através de configurações descritas em alto nível, chamadas “*recipes*”. As *recipes* contém informações sobre os pacotes que devem ser instalados, serviços que devem ser ativados e arquivos de configurações a serem editados. As máquinas a serem

configuradas podem ser agrupadas por funções (“roles”) de forma que o usuário seleciona quais funções um novo servidor deve ter e o Chef aplica as *recipes* necessárias para configurá-lo (OPSCODE, 2011).

Há uma grande diversidade de ferramentas para o gerenciamento de sistemas em rede, estas ferramentas estão divididas em quatro classes: protocolos nos quais padrões são definidos e implementados em hardware ou software para gerenciar dispositivos de rede (ex.: *Simple Network Management Protocol* - SNMP), linguagens de configuração que definem padrões para a definição de dados em um nível de abstração mais alto que os protocolos (ex.: Cfengine, Puppet, Chef), ferramentas para gerenciamento distribuído com as quais é possível gerenciar simultaneamente diversas máquinas a partir de um único ponto (ex.: Config4GNU (http://config4gnu.sourceforge.net) e as já mencionadas Cfengine e Puppet) e ferramentas para gerenciamento de uma única máquina que agilizam e simplificam o gerenciamento de uma única máquina (ex.: Webmin (http://www.webmin.com), Linuxconf (http://www.solucorp.qc.ca/linuxconf)) (KREUTZ, 2005, capítulo 2.3).

2.5 Resumo

Com o crescimento do número de empresas que utilizam virtualização em suas infraestruturas de TI cresce também a disputa entre empresas que desenvolvem soluções para o gerenciamento de virtualização. Esta disputa leva empresas como WMware, Citrix, Microsoft, Red Hat, dentre outras que estão neste mercado, a evoluírem constantemente seus produtos de forma que atualmente se equiparam em termos de recursos e funcionalidades.

A computação em nuvem, também tendência na computação, tem sido utilizada como forma de prover software, plataforma e infraestrutura como serviço de forma que recursos computacionais são disponibilizados por demanda remotamente (intranet, internet). Soluções específicas para o gerenciamento de nuvens computacionais tem sido desenvolvidas para facilitar e acelerar a criação de nuvens privadas, públicas ou híbridas (privadas e públicas).

Máquinas físicas e virtuais possuem sistemas e serviços com configurações específicas. Todo o processo de provisionamento, configuração e manutenção envolve custos operacionais que podem ser reduzidos pela utilização de ferramentas especificamente projetadas para este fim. Estas ferramentas são utilizadas em infraestruturas virtuais provendo, configurando e mantendo sistemas e serviços enquanto soluções de virtualização e gerenciadores de nuvens se encarregam do provisionamento e gerenciamento de máquinas virtuais.

Virtual appliances são máquinas virtuais com sistema operacional e serviços instalados, configurados e prontos para serem utilizados. *Virtual appliances* se tornaram uma forma de prover serviços onde todo o processo de provisionamento e configuração de máquinas virtuais, sistemas operacionais e serviços é substituído pelo processo de instanciação da *virtual appliance*.

O capítulo seguinte analisa a problemática do gerenciamento de máquinas virtuais e serviços utilizando as soluções apresentadas neste capítulo, especialmente as restrições encontradas em empresas de pequeno porte onde os recursos financeiros são limitados e não há pessoal especializado para a utilização destas ferramentas. Após esta análise este trabalho propõe uma abordagem que possibilite o gerenciamento integrado de máquinas virtuais e serviços com processos automatizados a partir de *templates*.

3 PROBLEMÁTICA E PROPOSTA

3.1 Desafios no gerenciamento de servidores virtuais e serviços de rede

Para a criação de uma máquina virtual é necessário que o usuário interaja com um *hypervisor*. Xen e KVM, por exemplo, possuem ferramentas CLI através das quais pode-se criar e gerenciar máquinas virtuais. Ferramentas e soluções de virtualização facilitam e agilizam este processo através de interfaces gráficas intuitivas. Embora estas ferramentas e soluções facilitem e agilizem o processo de criação de máquinas virtuais, o provisionamento de servidores virtuais ainda envolve as seguintes etapas:

1. Instalação do sistema operacional;
2. Configuração do sistema operacional;
3. Instalação dos *softwares* que compõe uma aplicação (serviço a ser executado pelo servidor);
4. Configuração dos serviços;
5. Testes.

Utilizando os recursos de provisionamento das soluções de virtualização pode-se reduzir parte do custo do processo de provisionamento de servidores virtuais. Uma vez que um servidor é criado, seu disco virtual e informações, como as referentes às configurações de hardware, podem ser utilizados como *template* para a criação de novos servidores virtuais. A eficiência da utilização de *templates* permite, às soluções de virtualização, a utilização do recurso da clonagem, desta forma as duas primeiras etapas dentre as citadas anteriormente são substituídas por um processo de cópia de uma máquina virtual existente (RED HAT, 3 min. e 40 seg.). Essa abordagem é utilizada por soluções de virtualização e gerenciadores de nuvens, como o OpenNebula, no qual discos de máquinas virtuais existentes podem ser utilizados como *templates* (OPENNEBULA PROJECT LEADS, 2011).

Soluções para o gerenciamento de virtualização e gerenciadores de nuvens vem sendo amplamente utilizados como forma de prover e gerenciar servidores virtuais de forma ágil e automática. Contudo, muitas empresas de pequeno e médio porte não dispõe de recursos e pessoal especializado para trabalhar com tais soluções. As

soluções de virtualização possuem planos de licenciamento complexos e de alto custo (VERAS, 2011, p. 13-15) restando às empresas menores utilizar as versões gratuitas de soluções de virtualização e gerenciadores de nuvens onde não estão disponíveis recursos como balanceamento de carga, alta disponibilidades, recuperação de desastres e outros que automatizam o gerenciamento de infraestrutura virtualizada de TI.

Ferramentas para o gerenciamento de serviços, como as apresentadas na seção 2.2, podem ser utilizadas para a instalação, configuração e manutenção de serviços juntamente com as soluções para virtualização em um mesmo conjunto de servidores virtuais, porém trabalhando de forma independente. Estas soluções focam ambientes de larga escala como Twitter, Oracle, Rackspace Hosting, Nokia que utilizam Puppet (PUPPET LABS, 2011), ou AMD, at&t, Cisco, Disney, Nasa e Facebook que utilizam Cfengine (CFENGINE, 2011), onde geralmente existem equipes de profissionais especializados no gerenciamento de serviços. Este é um cenário diferentes de pequenas e médias organizações, onde normalmente há falta de profissionais e baixo índice de qualificação, complicando o uso de soluções mais complexas e distribuídas.

Estas ferramentas, embora auxiliem na administração de sistemas e serviços, requerem conhecimento específico de suas linguagens de configuração para que sejam utilizadas corretamente e com maior proveito de seus recursos, o que pode levar administradores a preferir a manutenção manual em redes de pequeno porte. As soluções Cfengine, Puppet e Chef requerem conhecimento específico de suas linguagens de configuração, como pode ser visto no exemplo de código Puppet mostrado na figura 2 onde é definida uma classe do serviço NTP. O código mostrado como exemplo pode facilmente ser confundido com uma classe de programação orientada a objetos, porém uma classe Puppet deve ser compreendida como um papel, uma função a ser exercida por um sistema como parte de sua identidade (PUPPET, 2011).

Nas soluções de virtualização o provisionamento de serviços só é integrado ao processo de criação de máquinas virtuais com a utilização de *golden images* onde novas máquinas virtuais são clonadas a partir de *snapshots* de máquinas virtuais já existentes

com serviços implantados (CITRIX, 2010, p. 3; RED HAT, 3 min. e 50 seg.), o que pode ser considerado uma *virtual appliance*.

```
# ntp-class1.pp

class ntp {
  case $operatingsystem {
    centos, redhat: {
      $service_name = 'ntpd'
      $conf_file    = 'ntp.conf.el'
    }
    debian, ubuntu: {
      $service_name = 'ntp'
      $conf_file    = 'ntp.conf.debian'
    }
  }

  package { 'ntp':
    ensure => installed,
  }

  service { 'ntp':
    name      => $service_name,
    ensure    => running,
    enable    => true,
    subscribe => File['ntp.conf'],
  }

  file { 'ntp.conf':
    path    => '/etc/ntp.conf',
    ensure  => file,
    require => Package['ntp'],
    source  => "/root/learning-manifests/${conf_file}",
  }
}
```

Fonte: Puppet, 2011

Figura 2: Exemplo de código Puppet

Como visto no capítulo 2, *virtual appliances* são uma alternativa para reduzir os custos deste processo, pois são máquinas virtuais que encapsulam sistema operacional e serviços, porém não adaptadas ao contexto no qual serão executadas. Durante sua primeira execução é necessário configurar parâmetros de rede, nome de usuário e senhas ou até mesmo data e hora (VMWARE, 2007, p. 4). Sendo a aplicação um serviço de rede, neste caso deverá ser configurado para que atender às necessidades da rede. Adicionalmente, soma-se o fato de o serviço precisar de manutenção das configurações. Este processo, em aplicações tradicionais, é previsto para ser realizado de forma manual e pouco automatizada.

A diversidade da arquitetura de monitores de máquinas virtuais impõe mais um desafio à utilização de *virtual appliances* em ambientes heterogêneos (ambientes com diversos MMVs) (KREUTZ, 2009, p. 38). Trabalhos recentes propõem soluções para entrega e gerenciamento de *virtual appliances* em ambientes heterogêneos, como a proposta flexVAPs que propõe um sistema que facilita aos administradores a criação de *virtual appliances* e as entrega nas máquinas locais dos usuários. A proposta é baseada em *cache* e utiliza repositórios que contém componentes para a manutenção do sistema, armazenamento e para atender as restrições das *virtual appliances* impostas pela heterogeneidade do ambiente (KREUTZ, 2009).

Há também a proposta de um formato aberto para virtualização: o *Open Virtualization Format* (OVF). Segundo a VMware um formato aberto para a virtualização se justifica pela necessidade de um padrão na forma de encapsular e distribuir máquinas virtuais, a "VMware e outros líderes no campo da virtualização criaram o OVF, um formato de encapsulamento e distribuição para máquinas virtuais independente de plataforma, eficiente, extensível e aberto" (VMware, 2011, tradução nossa).

Este trabalho propõe uma solução que integre o gerenciamento de servidores virtuais e o gerenciamento de seus respectivos serviços baseando-se em descrições de sistemas e serviços, denominados *templates*, armazenados em um repositório. A proposta é utilizar um repositório central que contém *templates* de sistemas, aplicações e serviços. Além disso, ele contém a descrição atualizada das configurações e dados dos serviços. A partir desses dados tornar-se possível a entrega, inicialização e manutenção dinâmica e autônoma de serviços na rede.

A tabela 1 demonstra o escopo da solução proposta em relação às soluções para virtualização e ferramentas para o gerenciamento de serviços. A ideia desta abordagem é agilizar o provisionamento de servidores virtuais e a implantação e atualização de serviços de rede.

A criação de MVs, apresentada na tabela 1, refere-se à criação de uma sistema computacional virtualizado sobre um hardware virtual definido no momento da sua criação. O provisionamento de MVs consiste da criação de máquinas virtuais com sistema operacional instalado e, possivelmente, serviços instalados. Este processo normalmente é realizado através de clonagem.

A instalação, configuração e manutenção de serviços, neste contexto, refere-se à automatização deste processo por uma ferramenta de gerenciamento de sistemas e serviços.

A solução proposta neste trabalho abrange deste a criação de MVs até a atualização de serviços. Este trabalho integra, em uma solução, o gerenciamento de máquinas virtuais e serviços automatizando o provisionamento de servidores virtuais configurados e prontos para uso e a atualização de seus serviços.

TABELA 1

Abrangência das soluções e ferramentas de gerenciamento

	Criação de MVs	Provisionamento de MVs	Instalação de serviços	Configuração de serviços	Atualização de serviços
Soluções de virtualização	**X**	**X**			
Gerenciadores de nuvem	**X**	**X**			
Gerenciadores de serviço			**X**	**X**	**X**
Solução proposta	**X**	**X**	**X**	**X**	**X**

3.2 Solução proposta

A proposta deste trabalho consiste em integrar o gerenciamento de servidores virtuais e serviços através de uma solução que armazena as configurações de máquinas virtuais e aplicações/serviços em um repositório e a partir destes dados automatiza o processo de criação e atualização de servidores virtuais e suas aplicações. No contexto da solução proposta os servidores virtuais são criados a partir de *templates* de sistema (uma máquina virtual com sistema operacional instalado e configurado), *templates* de aplicação (um ou mais *templates* de serviços que compõe uma aplicação) e arquivos de configuração de serviços (não contidos nos *templates* de serviços porque as configurações são únicas para cada demanda). A partir dos dados contidos nos

templates utilizados para a criação de um servidor virtual, um *template* de sistema é criado automaticamente e poderá ser utilizado como base para a criação de outros servidores.

A atualização de aplicações consiste da atualização de pacotes que compõe um serviço e da atualização de arquivos de configuração de um serviço. As atualizações são automatizadas assim que um *template* de aplicação ou arquivo de configuração é editado no repositório.

Um *template* de aplicação é composto por um ou mais *templates* de serviços. Um *template* de aplicação, como de um servidor web, por exemplo, conteria os *templates* de serviços como Apache, PHP e MySQL. Um *template* de serviço contém as informações necessárias para automatizar a instalação dos serviços, as configurações dos serviços, específicas para atender a uma demanda, estão contidas nos arquivos de configuração.

A ideia desta proposta é centralizar o gerenciamento de servidores virtuais (máquinas virtuais e serviços) em uma única interface que permita ao usuário:

- Criar *templates* de sistema;
- Criar *templates* de aplicação;
- Criar servidores virtuais;
- Gerenciar o ciclo de vida de servidores virtuais;
- Gerenciar serviços através da edição de *templates* e de arquivos de configuração de cada serviço.

A figura 3 ilustra a arquitetura da abordagem proposta. Esta abordagem provê a preservação dos dados utilizados em *templates* e sua reutilização para a criação de novos servidores virtuais.

No repositório são armazenados os *templates*, arquivos de configuração e *snapshots*. O módulo de criação de novos servidores virtuais utiliza *templates* e arquivos de configuração do repositório para a criação de novos servidores virtuais. O módulo de atualizações acessa os *templates* de serviços, contidos nos *templates* de aplicações, e arquivos de configuração para atualizar os serviços ativos nos servidores virtuais. Os

snapshots possibilitam a rápida substituição de servidores virtuais que venham a apresentar problemas.

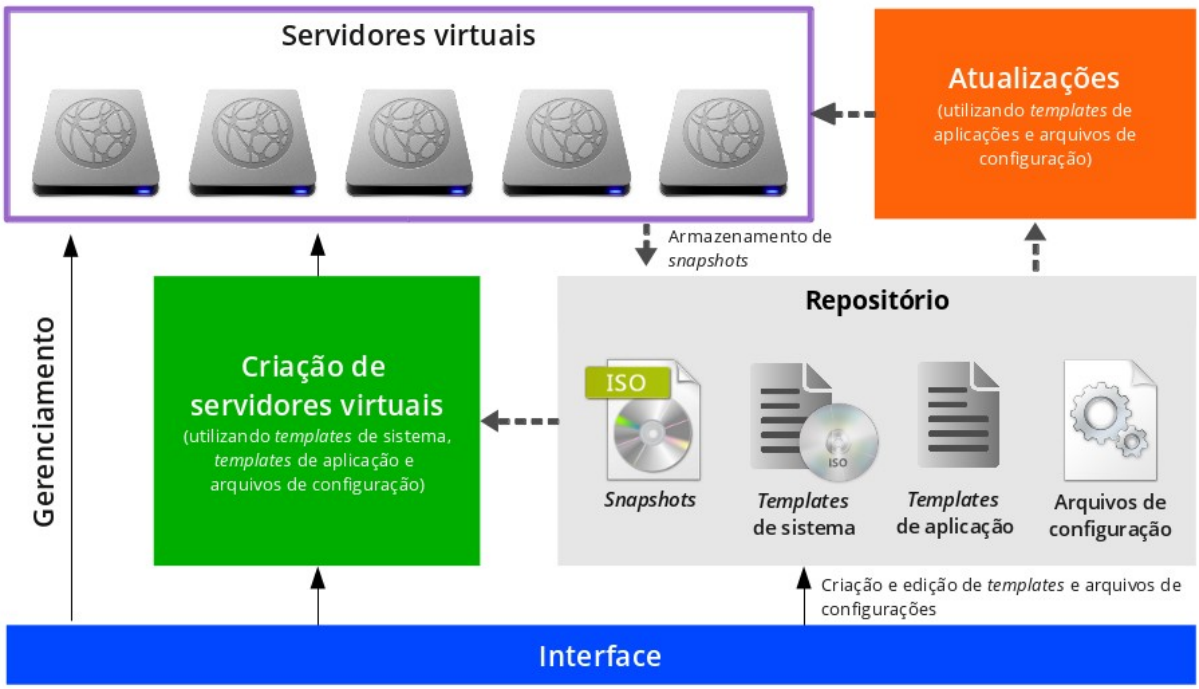


Figura 3: Arquitetura de um sistema de acordo com a abordagem proposta

O objetivo desta solução é agilizar e automatizar a criação de servidores virtuais com serviços automaticamente instalados e configurados, entregá-los prontos para uso como *virtual appliances* e automatizar a atualização de serviços.

3.2.1 Funcionamento e utilização

Esta seção descreve o funcionamento da solução proposta e a experiência de utilização por parte do usuário, sem entrar em detalhes técnicos. O objetivo desta seção é dar uma melhor compreensão do funcionamento do sistema e de sua utilização. A seguir serão apresentados os processos de criação de *templates*, criação de servidores virtuais e atualização de serviços.

Criação de *templates*: Os *templates* são compostos de conjuntos de informações de máquinas virtuais, serviços ou conjuntos de serviços que formam uma aplicação. Aos *templates* de máquinas virtuais (*template* de sistema) soma-se às informações um

arquivo imagem (disco virtual) de máquina virtual. Esta solução utiliza três tipos de *templates*: 1) sistema, 2) aplicação e 3) serviços.

1) *Template* de sistema: O procedimento consiste do fornecimento de informações sobre a máquina virtual e sistema operacional a ser utilizado (nome, descrição, detalhes do *hardware* virtual, SO) e a execução manual dos processos de instalação e configuração do sistema operacional na máquina virtual, de acordo com as regras e políticas da organização (se e quando for o caso). Um *template* de sistema bem configurado resultará em servidores virtuais potencialmente mais confiáveis e robustos. O processo de criação de *templates* de servidor é realizado uma única vez. O usuário, administrador do sistema, pode criar quantos *templates* de sistema forem necessários e/ou desejáveis. Cada *template* pode conter um novo sistema operacional, ou apenas configurações distintas. A figura 4 ilustra o processo de criação de um *template* de sistema.

Figura 4: Criação de *template* de sistema

2) *Template* de serviço: O usuário fornece informações sobre o serviço (nome, descrição), associa o *template* a um sistema operacional selecionando sistemas operacionais cadastrados no repositório e adiciona pacotes que devem ser instalados (inserindo os nomes de acordo com os disponíveis no repositório da distribuição Linux utilizada como sistema operacional). Este processo é ilustrado na figura 5.

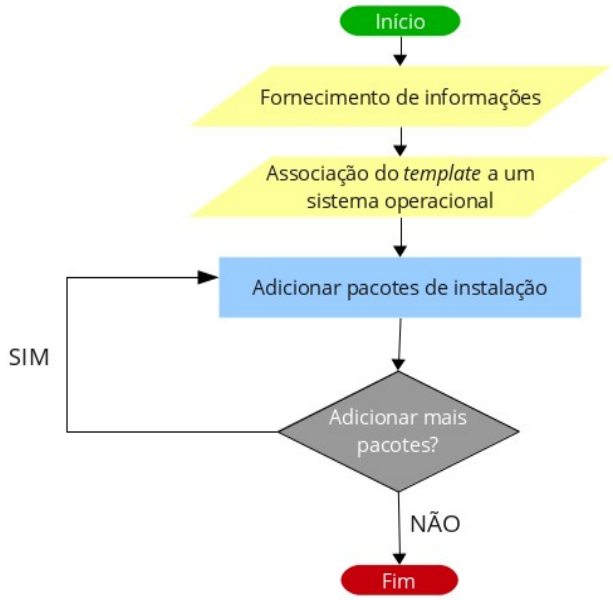


Figura 5: Criação de *template* de serviço

3) *Template* de aplicação: Os *templates* de aplicação são compostos por um ou mais *templates* de serviços previamente criados e armazenados no repositório. Ao usuário compete fornecer informações referentes à aplicação (nome, descrição), associá-la através da seleção a um sistema operacional e adicionar, também por seleção, os *templates* de serviços necessários. A figura 6 ilustra este processo.

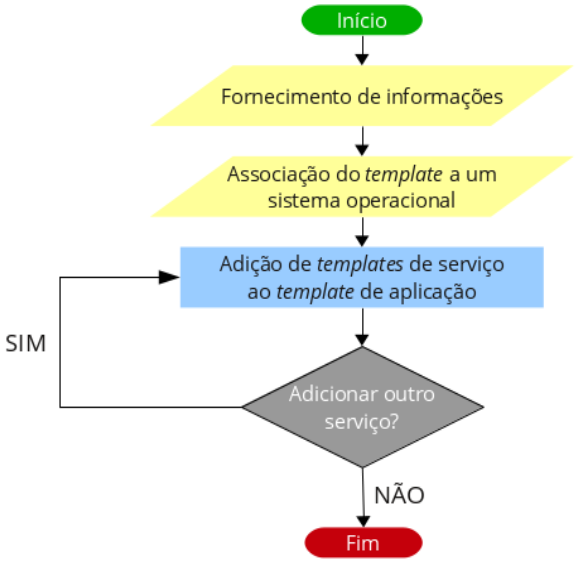


Figura 6: Criação de *template* de aplicação

Criação de servidores virtuais: A criação de novos servidores virtuais depende de *templates* de sistema, *templates* de aplicação e arquivos de configuração de serviços.

Os *templates* e arquivos de configuração estarão armazenados no repositório, podendo os arquivos de configuração de serviços serem adicionados ao repositório durante a criação de servidores virtuais.

Ao usuário, durante o processo de criação de um novo servidor virtual (ilustrado na figura 7), caberá fornecer informações como nome do servidor virtual, descrição e configurações de rede (endereço IP, máscara de rede, *gateway*), selecionar os *templates* de aplicações que deseja que sejam instaladas no novo servidor virtual e opcionalmente adicionar (através de seleção ou *upload*) e editar arquivos de configuração de serviços. Se optar por adicionar arquivos de configuração de serviços, o usuário deve fornecer informações referentes aos arquivos como nome e localização no sistema de arquivos do SO. Os procedimentos necessários para a criação do servidor virtual são automatizados pelo sistema.

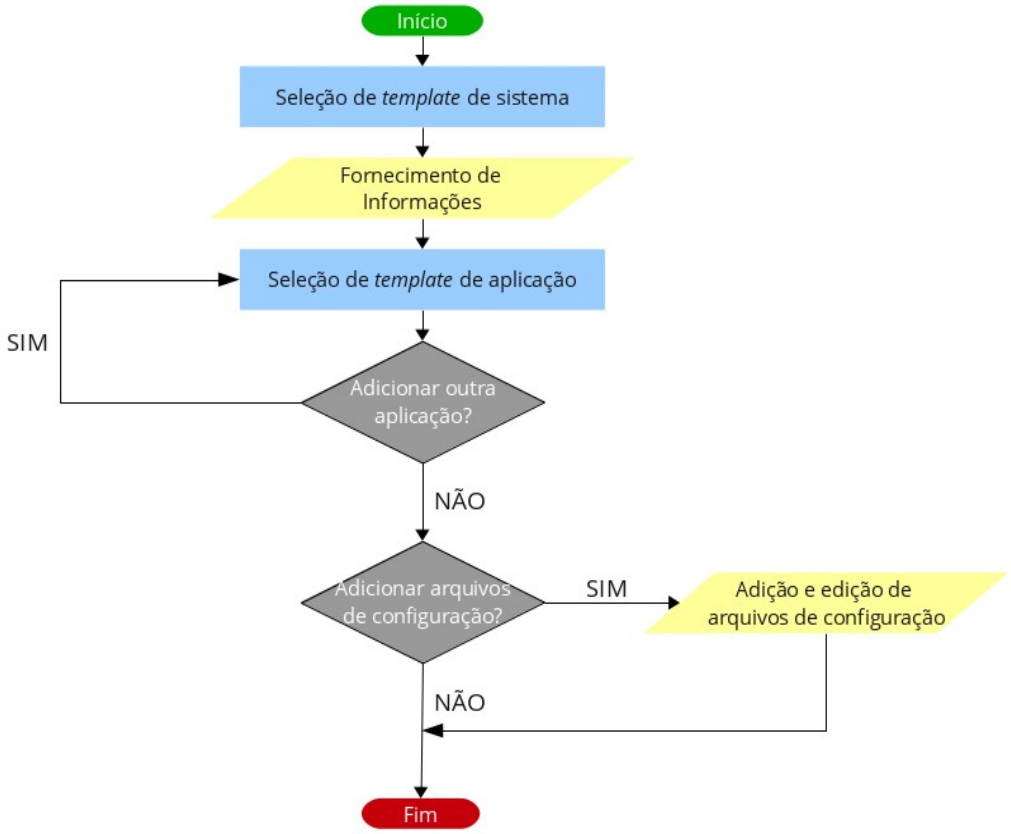


Figura 7: Criação de servidor virtual

Atualização de aplicações: As atualizações das aplicações ocorrem em duas situações: atualização da aplicação (atualização de pacotes de serviços) ou atualização das configurações de um serviço em um servidor virtual. Ambas automatizadas.

O primeiro caso de atualização ocorre quando um *template* de aplicação ou *template* de serviço é atualizado por meio de edição. O sistema propaga esta atualização para todos os servidores virtuais que utilizam o respectivo *template* editado.

No segundo caso, através do sistema, o usuário atualiza um arquivo de configuração de serviço que compõe uma aplicação e o sistema se encarrega de atualizar o serviço no servidor virtual.

Cenário de aplicação: A figura 8 ilustra um exemplo de rede de uma organização contendo 22 servidores, sendo 10 servidores físicos e 12 virtuais. Dentre os servidores físicos 4 são utilizados como hospedeiros de máquinas virtuais, cada um hospedando três servidores virtuais. Percebe-se que há diversos servidores com o mesmo serviço, porém com configurações diferentes.

O cenário apresenta instâncias de servidores virtuais executando os mesmos serviços. No exemplo cada uma das instâncias possui configurações diferentes para atender demandas distintas.

Este cenário reflete a realidade de empresas de pequeno e médio porte onde a administração é realizada nas condições descritas na seção 3.1. Em cenários como o ilustrado na figura 8, a solução proposta agiliza o provisionamento de servidores virtuais reutilizando *templates* e arquivos de configuração de serviços de servidores semelhantes. Os arquivos de configuração dos sistemas que executam em servidores físicos podem ser utilizados como base para a configuração dos serviços que serão instalados nos servidores virtuais. O provisionamento e as atualizações automatizadas, descritos nesta seção, proporcionam ganhos de tempo, confiabilidade e outros benefícios descritos a seguir.

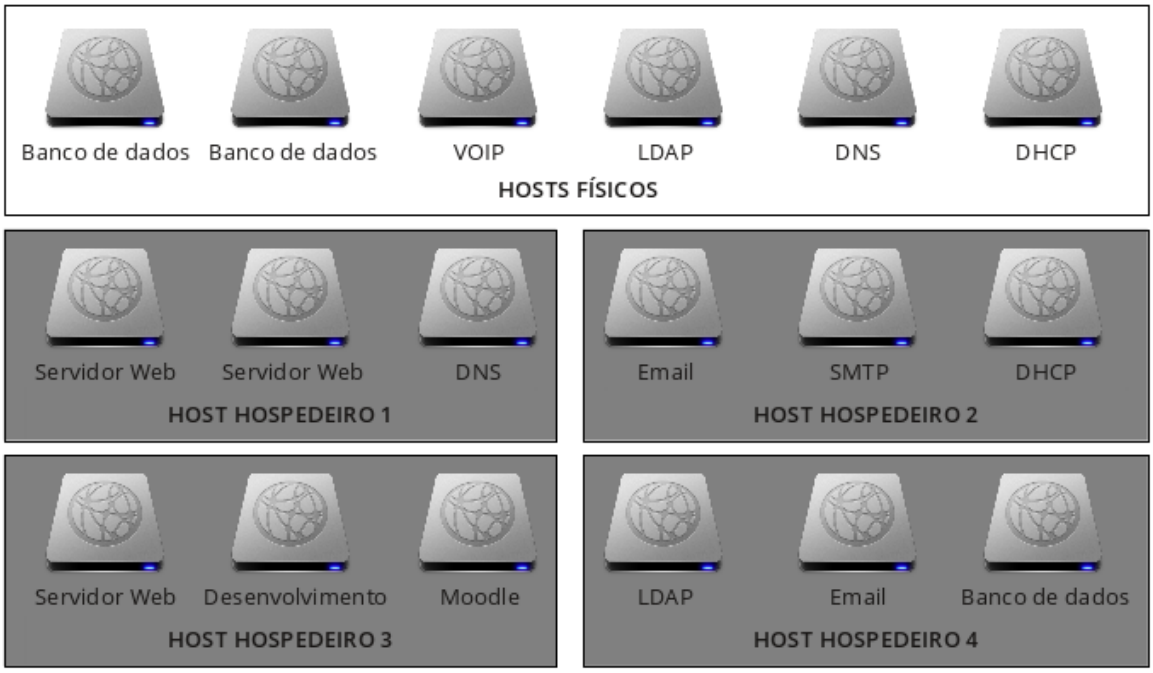


Figura 8: Exemplo de cenário de aplicação

Benefícios potencializados: Através da solução proposta é possível criar servidores selecionando *templates*, inserindo dados essenciais e adicionando ou editando arquivos de configuração de serviços. As informações armazenadas nos *templates* podem ser reaproveitadas. Um servidor virtual pode servir de base para a criação de outros servidores. Características como estas potencializam benefícios como:

- Agilidade no provisionamento de servidores virtuais;
- Agilidade na atualização de serviços;
- Reaproveitamento de informações de servidores virtuais e aplicações contidas em *templates*;
- Criação de novos servidores a partir de *templates* existentes, reduzindo a necessidade de conhecimentos mais avançados em sistemas e serviços
- Serviços providos através de *templates* sem intervenção manual;
- Confiabilidade: Característica potencializada pela automação nos processos de instalação e configuração de sistemas e serviços e pela utilização de *templates* que podem ser cuidadosamente elaborados e testados de forma que garantam um sistema e serviços mais confiáveis.

3.3 Resumo

Com a solução proposta o processo de criação de um servidor virtual é simplificado e automatizado, o que pode resultar em um processo mais rápido, potencializando mais agilidade no provisionamento e atualização de servidores virtuais. Os dados de configurações de máquinas virtuais e serviços são armazenados e reutilizados em novos *templates* e na criação de novos servidores virtuais. A atualização automatizada, baseada em configurações prévias, torna este processo mais confiável que processos manuais, pois este tem maior probabilidade de incorrer em erros.

4 IMPLEMENTAÇÃO E RESULTADOS

A arquitetura ilustrada na figura 3 e descrita no capítulo 3 está dividida em três módulos fundamentais: 1) repositório, 2) módulo de criação de servidores virtuais e 3) módulo de atualizações.

O objetivo deste capítulo é apresentar um protótipo de implementação da arquitetura proposta, bem como os resultados alcançados.

4.1 Repositório

No repositório são armazenados *templates*, imagens de máquinas virtuais (discos virtuais e *snapshots*) e arquivos de configuração de serviços. O repositório é composto de uma estrutura de diretórios e um banco de dados, conforme ilustrado na Figura 9. O banco de dados armazena as informações que compõe os *templates*, os diretórios armazenam imagens (discos virtuais) de servidores virtuais, *templates* de sistema e *snapshots*. Os *templates* de sistema são compostos por informações armazenadas no banco de dados e imagens armazenadas em diretórios.

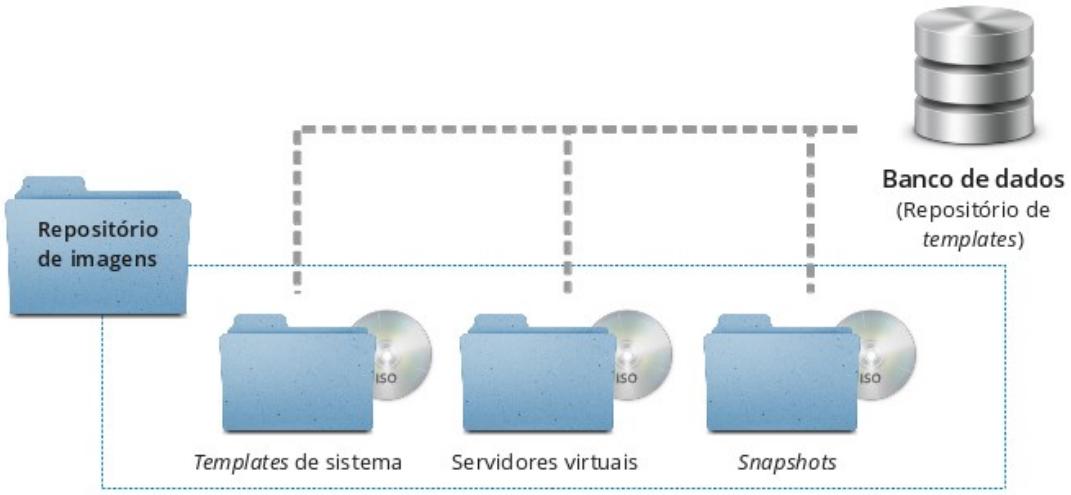


Figura 9: Ilustração do repositório implementado

4.1.1 Templates

Os *templates* definem serviços, aplicações e sistemas em conjuntos de informações independentes. A independência entre *templates* faz com que eles sejam reutilizáveis. Um *template* de serviço pode ser utilizado em diferentes *templates* de aplicação e um *template* de aplicação pode ser utilizado para a criação de diferentes servidores virtuais.

Informações como às referentes a SO são utilizadas em todos os *templates*, estas informações que se repetem e devem ser consistentes, pois as informações sobre o sistema operacional determinam se *templates* de sistemas, aplicações e serviços são compatíveis. Por esta razão foi utilizada a modelagem entidade-relacionamento para o armazenamento dos *templates* em um banco de dados SQL relacional.

Três tipos diferentes de *templates* são utilizados nesta solução, como apresentado na seção 3.2.1: 1) *template* de sistema, 2) *template* de aplicação e serviços e 3) *template* de servidor virtual. A seguir serão detalhados os três tipos de templates apresentando as informações que cada um contém e a modelagem entidade-relacionamento utilizada para o armazenamento das informações no banco de dados.

***Template* de sistema:** O t*emplate* de sistema resulta da criação e configuração de uma máquina virtual cujas informações fornecidas no momento da sua criação (figura 4) e sua imagem (disco virtual) serão utilizadas para criação de novos servidores virtuais.

Um *template* de sistema, como mostrado na tabela 2, contém um nome identificador, uma descrição que dê uma ideia das características do sistema, informações de identificação do SO instalado no *template*, informações sobre o *hardware* virtual e lista de serviços previamente instalados.

TABELA 2

Exemplo de *template* de sistema

Nome	vm1
Descrição	VM com instalação padrão do SO.
Tipo SO	Linux
Nome SO	Ubuntu
Versão SO	10.04 LTS
Arquitetura SO	x86_64
Número CPUs virtuais	1
Quantidade de memória	512MB
Tamanho de HD virtual	3GB
Lista de serviços instalados	SSH

Modelo entidade-relacionamento do *template* de sistema: No modelo entidade-relacionamento do *template* de sistema, ilustrado na figura 10, as informações sobre sistema operacional e a lista de serviços instalados, apresentados na tabela 2, são armazenados em tabelas separadas, como pode ser observado através do Anexo 1. O *template* de sistema apenas faz referência às chaves estrangeiras das entradas armazenadas nas tabelas de origem da descrição de SO, por exemplo (campo *id_operating_system* na figura 10).

Sistemas operacionais e serviços instalados são associados a um *template* de sistema através de relações entre entidades. Desta forma, por exemplo, é possível assegurar que um *template* de aplicação foi desenvolvido para o mesmo SO de um template de sistema, pois ambos os *templates* estão relacionados com o mesmo SO registrado na tabela de sistemas operacionais. Os serviços instalados podem ser mais de um, por esta razão são armazenados em uma tabela que armazena as relações do *template* de sistema e a tabela de lista de serviços instalados (tabela *system_template_services_list* na figura 10).

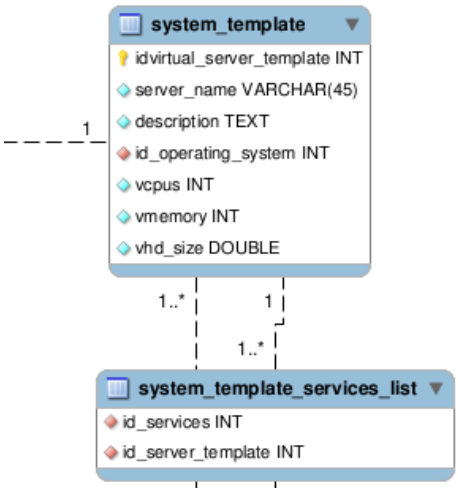


Figura 10: Modelo entidade-relacionamento de *template* de sistema

***Template* de serviço e *template* de aplicação :** Uma aplicação pode conter um ou mais serviços. Exemplos: 1) uma aplicação para atribuição de endereços de rede (DHCP) contém apenas um serviço, o DHCP, 2) um servidor web contém serviços como PHP, Apache2, MySQL e outros, dependendo da necessidade da aplicação. Uma aplicação como um servidor Wordpress reaproveita os *templates* de serviços utilizados no *template* da aplicação servidor web e adiciona um serviço, o Wordpress.

TABELA 3

Exemplo de *template* de serviço (DHCP)

Nome	DHCP_1
Descrição	Serviço DHCP para atribuição dinâmica de configurações de rede
Tipo SO	Linux
Nome SO	Ubuntu
Versão SO	10.04 LTS
Arquitetura SO	x86_64
Lista de pacotes a serem instalados	dhcp3-server

Um *template* de serviço contém, como demonstrado na tabela 3, um nome identificador, uma descrição do que o serviço é e o que faz, informações sobre o

sistema operacional e uma lista de pacotes que devem ser instalados para o funcionamento do serviço.

Os *templates* de aplicação devem conter informações referentes ao próprio *template* e aos serviços que o compõem. Configurações específicas da demanda do serviço (endereço de rede, arquivos de configuração) devem estar relacionadas com o servidor virtual que utiliza as aplicações e não à aplicação, desta forma um *template* de aplicação pode ser utilizado para a criação de diferentes servidores virtuais.

O *template* de aplicação, exemplificado na tabela 4, contêm um nome identificador, informações relativas ao sistema operacional, que garantem que todos os serviços de uma aplicação sejam compatíveis com o mesmo SO, e a lista de *templates* de serviços que são utilizados na aplicação.

TABELA 4

Exemplo de *template* de aplicação (DHCP)

Nome	Servidor DHCP
Tipo SO	Linux
Nome SO	Ubuntu
Versão SO	10.04 LTS
Arquitetura SO	x86_64
Lista de templates de serviços	DHCP_1

Modelo entidade-relacionamento dos *templates* de serviço e de aplicação: *Templates* de serviço e de aplicações são armazenados em tabelas distintas que se relacionam, como mostrado na figura 11. Esta relação é feita através da tabela *app_template_has_service* que relaciona um *template* de aplicação a um ou mais *templates* de serviço e um *template* de serviço a um ou mais *templates* de aplicação.

Os pacotes são armazenados na tabela *package* e são relacionados a um serviço através da tabela *service_has_packages*.

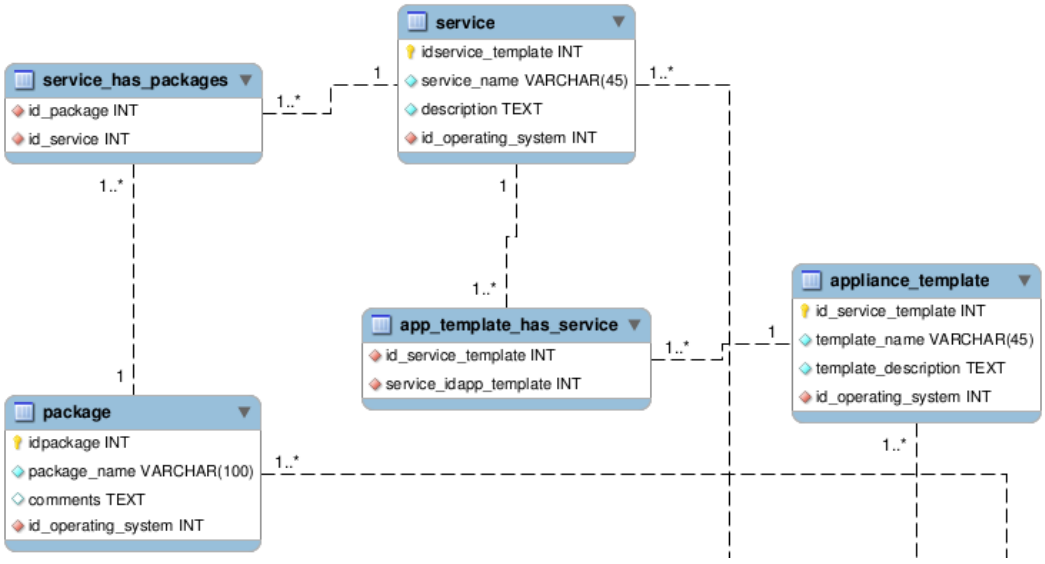


Figura 11: Modelo entidade-relacionamento de *template* de aplicação e *template* de serviços

As informações relativas ao sistema operacional são provenientes das mesmas tabelas mencionadas na modelagem entidade-relacionamento de *template* de sistema (Anexo 1). Nas tabelas de *template* de aplicação, serviço e na tabela de pacotes o SO é identificado através de uma chave estrangeira.

***Template* de servidor virtual:** Este *template* é semelhante ao *template* de sistema, porém contém algumas informações que são específicas de cada servidor virtual, como endereço de rede, lista de *templates* de aplicação que foram utilizados no durante o processo de criação do servidor virtual e arquivos de configuração dos serviços. Mais detalhes podem ser vistos na tabela 5.

Um *template* de servidor virtual contêm um nome identificador do template, uma definição de sua finalidade, informações sobre sistema operacional, informações sobre o hardware virtual, configurações de rede, lista das aplicações utilizadas no servidor virtual, lista de serviços previamente instalados (serviços instalados no *template* de sistema utilizado para a criação do servidor virtual) e lista de arquivos de configuração de serviços.

TABELA 5

Exemplo de *template* de servidor virtual

Nome	Servidor DHCP rede 192.168.0.2
Descrição	Servidor DHCP configurado para atender a demanda da rede 192.168.2.0/24
Tipo SO	Linux
Nome SO	Ubuntu
Versão SO	10.04 LTS
Arquitetura SO	x86_64
Número CPUs virtuais	1
Quantidade de memória	512MB
Tamanho de HD virtual	3GB
Endereço IP	192.168.1.40
Máscara de rede	255.255.255.0
Gateway	192.168.1.1
Endereço IP 2	192.168.2.2
Máscara de rede 2	255.255.255.0
Gateway 2	
Lista de aplicações	Servidor DHCP
Lista de serviços previamente instalados	SSH
Arquivos de configuração de serviços	dhcpd.conf, dhcp3-server

Modelo entidade-relacionamento do *template* de servidor virtual: As informações referentes aos *templates* de aplicação, configurações de rede e lista de serviços previamente instalados são armazenadas em tabelas distintas (Figura 12) e relacionadas ao *template* de servidor virtual através das tabelas, respectivamente,

virtual_server_has_services, *virtual_server_has_network* e *virtual_server_services_list*. As informações do sistema operacional são armazenadas uma tabela referenciada através da chave estrangeira *id_operating_system*, da mesma forma que nos *templates* de sistema, serviço e aplicações.

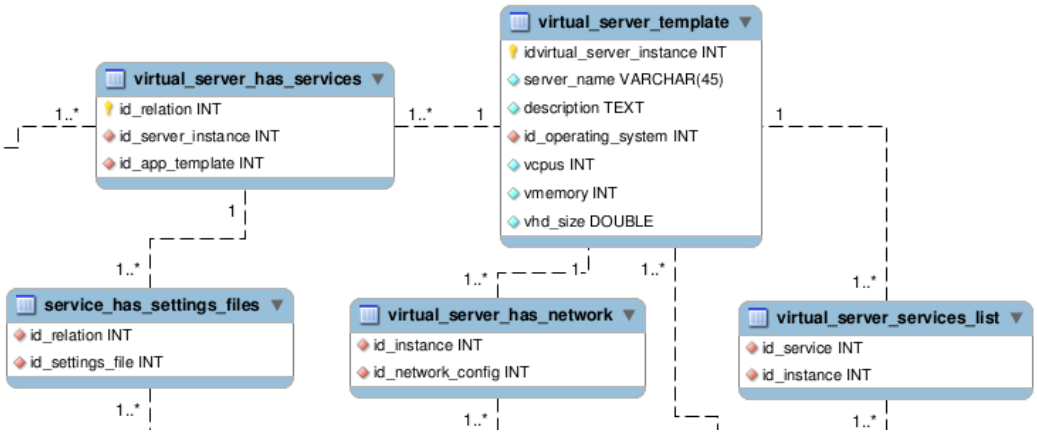


Figura 12: Modelo entidade-relacionamento de *template* de servidor virtual

Um servidor virtual pode estar relacionado a diversos *templates* de aplicação (relação feita através da tabela *virtual_server_has_services*). Os arquivos de configuração são associados a esta relação através da tabela *service_has_settings_files*.

4.2 Criação de servidores virtuais

A seção 3.2.1 trata da experiência do usuário na criação de um novo servidor virtual, processo que pode ser sintetizado em seleção de *templates*, fornecimento de informações (nome, descrição e configurações de rede) e adição de arquivos de configuração de serviços. A partir dos *templates* selecionados e dos arquivos de configuração de serviços adicionados, o sistema automatiza o processo de criação de um novo servidor virtual, como ilustrado na figura 13.

Este processo consiste de:

1. Clonagem do *template* de sistema selecionado;
2. Configuração de *hostname* e interfaces de rede do clone criado;
3. Instalação, no clone criado, das aplicações cujos *templates* foram selecionados;
4. Configuração das aplicações através dos arquivos de configuração adicionados.

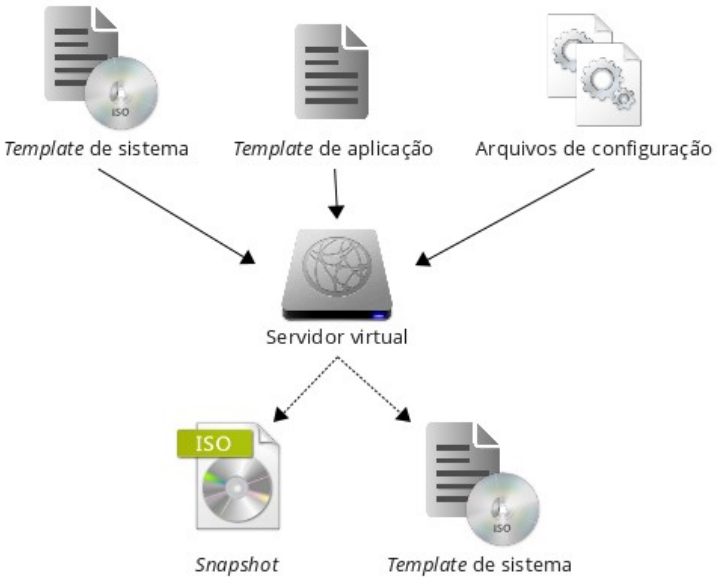


Figura 13: Criação de novo servidor virtual baseada em *templates* e arquivos de configuração de serviços

O processo de criação de uma nova máquina virtual, feito através da clonagem da imagem do *template* de sistema selecionado, é executado pela ferramenta Virt-clone (seção 4.4). Na imagem da nova máquina virtual os arquivos de configuração de rede e *hostname* são editados e *scripts* para a instalação dos serviços são criados a partir das informações contidas nos *templates* de serviço. Quando a máquina virtual é iniciada as alterações feitas na imagem tem efeito e o *script* de instalação de serviços é executado. Em seguida os arquivos de configuração de serviços, adicionados no processo de criação do servidor virtual (seção 3.2.1), são enviados para a máquina virtual que então é reiniciada. Quando este processo é concluído o novo servidor virtual está em execução com serviços instalados e configurados.

A partir das informações dos *templates* utilizados para a criação do novo servidor virtual e da sua imagem, um novo *template* de sistema é criado contendo todas as aplicações instaladas no servidor virtual e este *template* é disponibilizado para a criação de outros servidores. Assim que o processo de criação do servidor virtual é finalizado um *snapshot* é criado e armazenado no repositório para agilizar o processo de substituição do servidor virtual se for necessário.

4.3 Atualização de aplicações e serviços

As atualizações de aplicações ocorrem mediante a edição de *templates* de aplicação, de *templates* de serviço e de arquivos de configuração de serviços, como descrito na seção 3.2.1.

Um *template* de aplicação é composto por um ou mais *templates* de serviço. Se um *template* de serviço é atualizado ou outro *template* de serviço é adicionado ao *template* de aplicação, todos os servidores virtuais que o utilizam o *template* de aplicação alterado devem receber a atualização, como ilustrado na figura 14. Quando o usuário conclui as modificações em um *template* de aplicação através do sistema, o sistema verifica quais serviços foram adicionados ou removidos e executa as operações através da execução de comando remoto via SSH.

As aplicações são atualizadas nos servidores virtuais através da execução remota de comandos via SSH (seção 4.4). No protótipo desenvolvido as atualizações das aplicações se restringem a instalação e remoção de pacotes de serviços cujos *templates* foram editados.

Figura 14: Atualização de aplicações baseada na edição de *templates*

Os arquivos de configuração de serviços não estão relacionados ao *template* de aplicação ou *template* de serviço, como visto na seção 4.1.1, e sim ao servidor virtual que utiliza os arquivos de configuração de serviços para atender uma demanda específica. Logo, atualização de um arquivo de configuração afeta apenas o servidor virtual que o utiliza, como mostrado na figura 15.

A edição dos arquivos de configuração é feita através da solução desenvolvida. A importância de editar e atualizar um arquivo de configuração através da solução, e não diretamente no servidor virtual, está na proposta de reutilização de arquivos deste sistema. Desta forma se um servidor igual ou idêntico precisa ser criado os arquivos de configuração podem ser reaproveitados.

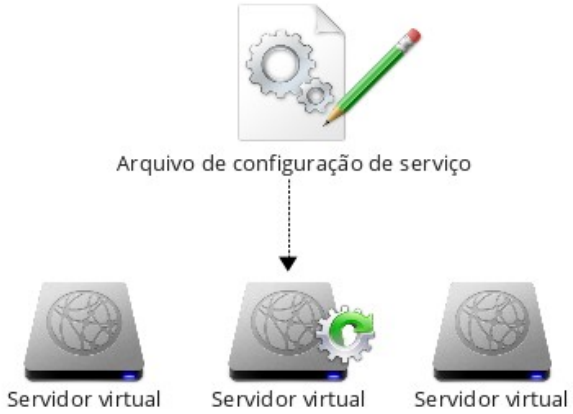


Figura 15: Atualização de serviços baseada na edição de arquivos de configuração

Quando a edição de um arquivo de configuração de serviço, armazenado no repositório, é concluída por parte do usuário, a solução automaticamente atualiza o arquivo no servidor virtual correspondente. A solução utiliza a ferramenta SCP, contida no SSH, que permite a transferência de arquivos através de uma conexão SSH. As informações referentes ao arquivo, como a localização no sistema de arquivos do SO, são obtidas da tabela do banco de dados onde os arquivos são armazenados (tabela *settings_file*, Anexo 1).

4.4 Recursos utilizados para implementação

Ubuntu GNU Linux: GNU Linux é o sistema operacional utilizado em 60% dos servidores web em todo o mundo, afirmou Steve Ballmer, CEO da Microsoft (NICCOLAI, 2008). Na última lista dos sites de hospedagem mais confiáveis feita pela Netcraft (http://news.netcraft.com/), ao menos 25 dos 41 melhores sites rodam sobre Linux (4 dos sites listados não tem sistema operacional identificado) (NETCRAFT, 2011). Em dados estatísticos de junho de 2011, levantados pela Top 500 *Supercomputer Sites*

(http://www.top500.org), Linux roda em 91% dos primeiros 500 supercomputadores (TOP500.Org, 2010).

Devido a sua modularidade, sua performance, seu poder de eficiência, sua ampla utilização, ubiquidade e pelo fato de ser *open source*, Linux é o sistema operacional para computação em nuvem, afirma McPherson (2009).

Ubuntu é a distribuição Linux mais popular para *desktops* de acordo com o site DistroWatch (http://distrowatch.com/). O Ubuntu é fruto do desejo do empresário sul africano Mark Shuttleworth de se fazer um sistema operacional Linux para *desktops* de fácil utilização. O Ubuntu atualmente é mantido pela empresa Canonical Ltd (CANONICAL, 2011).

Para a implementação deste sistema a versão utilizada é 10.04 LTS *server*. Apenas este sistema operacional foi utilizado tanto para o servidor hospedeiro quanto para os servidores virtuais.

***Kernel Based Virtual Machine* – KVM:** é uma solução de virtualização *open source* para arquitetura x86. Consiste de um módulo que é carregado no *kernel* Linux que provê a uma infraestrutura de virtualização e módulos específicos para processadores com as tecnologias Intel VT e AMD-V. KVM é uma solução de virtualização total que suporta sistemas operacionais não modificados. (RED HAT EMERGING TECHNOLOGIES).

KVM é uma tecnologia ascendente que vem ganhando espaço no mercado de virtualização. Em 2010 a Red Hat adotou o KVM como sua principal solução de virtualização e sobre o qual suas soluções serão desenvolvidas por ser, segundo a Red Hat, “a próxima geração de tecnologias de virtualização” (RED HAT, 2010). Recentemente as empresas Red Hat, HP, IBM, Intel, Eucalyptus, BMCSoftware e SUSE anunciaram a *Open Virtualization Alliance* (http://www.openvirtualizationalliance.org/index.html) com o objetivo de incentivar e promover soluções para virtualização baseadas no KVM.

Neste trabalho foi utilizada a versão 0.12.5.

Libvirt: é uma API para virtualização com suporte aos hypervisors KVM, Xen, VMware, OpenVZ, dentre outros. Provê interface para gerenciamento local ou remoto

de máquinas virtuais, redes virtuais e armazenamento (RED HAT EMERGING TECHNOLOGIES).

Neste trabalho foi utilizada a versão 0.8.3. Foi aplicada para controle do ciclo de vida das máquinas virtuais.

Virt-clone: é uma ferramenta para a clonagem de máquinas virtuais. Clona máquinas virtuais inativas, copia imagens de discos de máquinas virtuais, define as configurações como UUID e endereço MAC para o gerenciamento com a Libvirt. (RED HAT EMERGING TECHNOLOGIES).

Neste trabalho foi utilizada a versão 0.500.3. Foi utilizada na criação de novas virtual appliances.

Boost C++ Libraries: É um conjunto de bibliotecas modernas para C++ mantida por um grupo de desenvolvedores criado em 1998 após o lançamento da primeira revisão do padrão C++ (C++98). As bibliotecas Boost C++ aumentam a produtividade e aceleram o desenvolvimento de projetos por oferecer diversas funcionalidade que possibilitam a escrita de código em C++ moderno (SCHÄLING, 2010).

Neste trabalho foi utilizada a versão 1.42.

O sistema foi desenvolvido em C++ utilizando recursos de bibliotecas Boost, como por exemplo, para manipulação de strings.

PostgreSQL: é um sistema de banco de dados SQL relacional apresentado, na própria página do projeto, como “o mais avançado banco de dados *open source* do mundo”. Tem sido desenvolvido ativamente por mais de 15 anos e sua arquitetura é conhecida por prover confiabilidade, integridade de dados, dentre outras características (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2011, tradução nossa).

Neste trabalho foi utilizada a versão 8.4.

Este banco de dados foi utilizado para armazenamento dos *templates*.

Secure Shell (SSH): é um programa que possibilita acesso, execução de comandos, transferência de arquivos em máquinas remotas através de uma conexão

segura encriptada entre máquinas desconhecidas e através de redes inseguras (CHRISTIAS, 1994).

Neste trabalho foi utilizada a versão 1:5.5.

A comunicação entre o servidor hospedeiro e as *virtual appliances* e a transferência de arquivos é realizada através de SSH.

4.5 Resultados

Através da solução implementada alcançou-se o objetivo de integrar o gerenciamento de máquinas virtuais e serviços no que se refere ao provisionamento de servidores virtuais e atualização de aplicações. A utilização de *templates* como forma de definir servidores virtuais, serviços e aplicações simplificou o processo de criação de servidores virtuais para o usuário de forma que a solução automatiza o processo de provisionamento de servidores virtuais baseando-se nos *templates* selecionados, informações fornecidas e arquivos de configuração de serviços, como apresentado na seção 4.2.

Para o provisionamento de servidores virtuais, quando realizado manualmente, conforme visto na seção 3.1, o processo implica em:

- criação da máquina virtual através de uma ferramenta CLI ou GUI;
- instalação do SO;
- configuração do SO;
- instalação de serviços;
- configuração de serviços
- testes.

Utilizando a solução proposta neste trabalho, todas as etapas acima são suprimidas. O *template* de sistema utilizado para a criação do servidor virtual já contém *hardware* virtual definido e SO instalado e configurado. A instalação e configuração de serviços é feita automaticamente pelo sistema utilizando dados armazenados em *templates* de serviços. Se os *templates* de sistema e serviços contiverem configurações

previamente testadas, o novo servidor virtual criado a partir dos *templates* não necessita testes.

As soluções com recursos de provisionamento de servidores virtuais, através da utilização de *templates* ou *virtual appliances*, não oferecem recursos para que os serviços sejam gerenciados. O provisionamento, porém, suprime as etapas de criação de MV e instalação e configuração de SO. Em se tratando de *virtual appliances* a etapa de instalação de serviços também é suprimida.

O processo de criação de servidores virtuais, utilizando a solução proposta, consiste de:

- seleção do *template* de sistema;
- seleção do *template* de aplicação;
- inserção de informações de configuração de rede;
- inserção ou seleção de arquivos de configuração de serviços;

A solução proposta neste trabalho diferencia-se das soluções de virtualização e gerenciadores de nuvem apresentados no capítulo 1 por integrar o gerenciamento de máquinas virtuais e serviços, como pode ser observado na tabela 6. As informações armazenadas em *templates* permitem a automatização do provisionamento de servidores virtuais com serviços instalados e configurados para atender demandas específicas. As informações armazenadas em *templates* de aplicação possibilitam a atualização de serviços rodando em servidores virtais através da mesma solução.

Para a realização deste trabalho utilizou-se *templates* previamente armazenados no banco de dados que compõe o repositório, cujas tabelas foram populadas através do *script* inicial.sql (Anexo 2), e uma imagem de MV, armazenada em diretório do repositório, com a instalação padrão do SO Ubuntu GNU Linux como parte do *template* de sistema. Utilizou-se dois *templates* de aplicação: 1) servidor DHCP, composto pelo *template* do serviço DHCP, e 2) *template* de servidor Web, composto pelos *templates* dos serviços MySQL, PHP e Apache2. Com base nos *templates* armazenados no repositório a criação de um novo servidor virtual pôde ser realizada conforme a proposta apresentada na seção 3.2.

TABELA 6

Demostrativo das etapas automatizadas nos diferentes processos de provisionamento de servidores virtuais

Etapa	**Método de Provisionamento**			
	Manual	Semiautomatizado (utilização de *templates*)	virtual appliances	Automatizado (solução proposta)
Criação de VM		**X**	**X**	**X**
Instalação de SO		**X**	**X**	**X**
Configuração de SO		**X**	**X**	**X**
Instalação de serviços			**X**	**X**
Configuração de serviços				**X**
Atualização				**X**

Para o cálculo de tempo estimado da criação de servidor virtual foi utilizado o *template* de sistema descrito no parágrafo anterior, o *template* de aplicação de servidor DHCP, dois arquivos de configuração de servidor DHCP e configurações de rede previamente definidas. O mesmo servidor DHCP foi criado 10 vezes com exatamente as mesmas configurações utilizando a solução implementada. O processo de criação do servidor DHCP consistiu do seguinte processo:

- seleção do *template* de sistema (figura 16);
- fornecimento das informações: nome do servidor, descrição do servidor;
- adição de duas placas de rede virtuais e inserção das configurações de endereço de rede para as duas placas;
- seleção do *template* de aplicação do servidor DHCP;
- adição dos arquivos de configuração do servidor DHCP *dhcpd.conf* e *dhcp3-server* que estavam armazenados no disco local, porém não no repositório da solução.

O tempo médio aproximado, obtido através da criação do servidor DHCP realizada 10 vezes, foi de 4 minutos e 40 segundos contados a partir da inicialização da solução até o ponto em que o servidor virtual está pronto e em execução. O tempo total

de cada execução foi obtido subtraindo-se o tempo inicial (inicialização da solução) do tempo final (servidor virtual está pronto e em execução). No tempo total estimado está incluso o tempo médio aproximado de interação com o usuário, 1 minuto e 10 segundos.

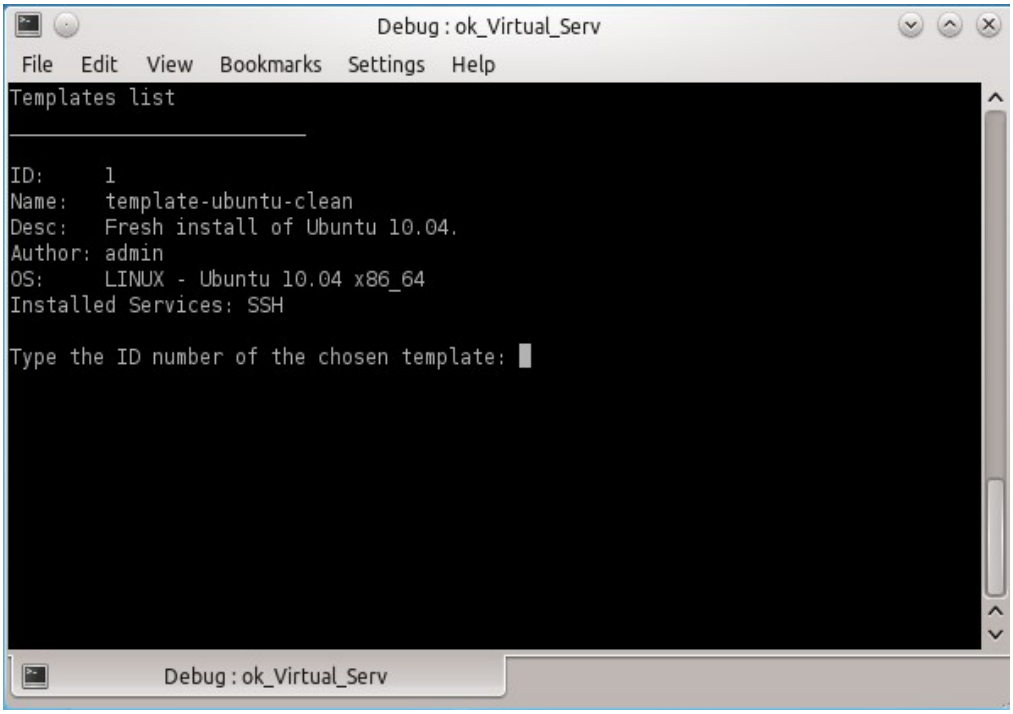


Figura 16: Seleção de *template* de sistema

Esta proposta conseguiu definir servidores virtuais, aplicações e serviços em *templates* de forma independente. Esta independência permite que o conjunto de informações de um *template* seja modificado sem que as modificações interfiram nos demais *templates*. Os *templates* foram definidos em conjuntos de informações utilizados para o provisionamento de servidores virtuais e atualização de serviços. Outras informações podem ser adicionadas aos *templates* de forma que outros recursos possam ser implementados tornando o gerenciamento de servidores virtuais e serviços mais abrangente através de novas funcionalidades, por exemplo, em um *template* de serviço poderia constar informações sobre os diretórios e arquivos de um serviço e o sistema conferiria a existência e as permissões destes diretórios e arquivos com base nas informações contidas no *template*.

Quando um servidor virtual é criado a partir de *templates,* o servidor virtual segue relacionado a esses *templates*. Esta relação possibilita que as atualizações feitas em *templates* armazenados no repositório sejam replicadas a todos os servidores virtuais criados a partir dos *templates* editados. Esta solução realiza atualizações automáticas em duas situações:

1) Quando um ou mais *templates* de serviços são adicionados a um *template* de aplicação, todos os servidores com aplicações instalados a partir do *template* atualizado recebem as atualizações. Por exemplo, se ao *template* de servidor Web composto pelos *templates* de serviço MySQL, PHP e Apache2 for adicionado o *template* de serviço Wordpress, o Wordpress será instalado nos servidores criados a partir do *template* de aplicação servidor Web;

2) Quando um arquivo de configuração de serviço é editado no repositório, o servidor virtual que utiliza este arquivo de configuração para a configuração dos seus serviços recebe automaticamente a versão editada do arquivo de configuração. Por exemplo, se o arquivo *dhcpd.conf* utilizado na criação do servidor virtual DHCP for editado, o arquivo *dhcpd.conf* do servidor virtual será substituído pela versão do arquivo editada no repositório.

A edição de *templates* e arquivos de configuração em um repositório central possibilita que *templates* e arquivos de configurações sejam reutilizados. Para a obtenção do tempo médio estimado para a criação de um servidor virtual DHCP, dez servidores virtuais foram criados utilizando os mesmos arquivos. Se, por exemplo, as configurações do serviço fossem alteradas no arquivo *dhcp.conf* para cada um dos dez servidores virtuais criados e, para cada servidor fossem atribuídos endereços de rede diferentes, teríamos dez servidores virtuais distintos utilizando o mesmo *template* de sistema, o mesmo *template* de aplicação e o mesmo arquivo *dhcp3-server*.

Os códigos desenvolvidos para este trabalho estão disponíveis no endereço https://github.com/arbiza/Virtual-Services.

5 CONCLUSÃO

A virtualização, afirmada como uma das principais tendências na computação por diversos autores, está em constante evolução. As soluções desenvolvidas para gerenciar infraestruturas virtuais de TI e nuvens computacionais acompanham esta evolução oferecendo recursos e funcionalidades que possibilitam o gerenciamento ágil, centralizado e autônomo de conjuntos de máquinas virtuais, armazenamento e redes virtuais.

Em ambientes desse gênero o gerenciamento de serviços encontra desafios devido a possível diversidade de sistemas, serviços e demandas a serem administradas. Com isso ferramentas surgiram para facilitar e automatizar a instalação, configuração e manutenção de sistemas e serviços.

Os recursos das soluções de virtualização que possibilitam o gerenciamento mais autônomo e mais completo da infraestrutura de TI virtualizada só são disponibilizados através da aquisição de licenças. As ferramentas para o gerenciamento de serviços possuem linguagens próprias que requerem especialização para que sejam utilizadas. Organizações de pequeno e médio porte não possuem recursos e nem pessoal disponível para que possam se beneficiar dos recursos oferecidos por estas soluções e ferramentas.

Este trabalho apresentou um protótipo de arquitetura de uma solução onde o gerenciamento de servidores virtuais e de serviços de rede é integrado em uma única solução que pode ser utilizada sem a necessidade de especialização. Esta solução automatiza o provisionamento de servidores virtuais utilizando informações armazenadas em *templates*. Para o usuário o processo de criação de servidores virtuais se resume a selecionar *templates*, fornecer informações referentes ao servidor virtual e adicionar arquivos de configuração de serviços, conforme descrito no capítulo 4. Com base nas informações de *templates* de aplicação e serviços foi possível automatizar a atualização de serviços em execução em servidores virtuais.

A solução implementada fornece as funcionalidades para o provisionamento e atualização de servidores virtuais e serviços baseado em templates, o que serve para demonstrar a viabilidade da proposta de gerenciamento integrado de máquinas virtuais

e serviços apresentada neste trabalho. A solução automatiza a criação de servidores virtuais com aplicações instaladas e configuradas e a atualização de serviços demonstrando que a abordagem apresentada é eficaz para gerenciamento de servidores virtuais no que se refere ao provisionamento e atualização de serviços. Com isso este trabalho alcançou o objetivo proposto.

Este trabalho apresenta resultados referentes ao provisionamento e atualização de servidores virtuais de forma automatizada. Para trabalhos futuros, dando sequencia ao desenvolvimento deste projeto, há ideias como: a implementação das funcionalidades que permitem controlar o ciclo de vida dos servidores virtuais, a adição de informações aos *templates* de serviços possibilitando agregar funcionalidades à solução como a checagem de permissões de diretórios e arquivos dos serviços, ampliar o suporte desta solução a outras distribuições Linux e a criação de uma interface web de gerenciamento.

6 REFERÊNCIAS

BAUER, Kirk. Automating Security with GNU Cfengine. **Linux Journal**. Fevereiro de 2004. online. Disponível em: <http://www.linuxjournal.com/article/6848?page=0,0>. Acesso em Maio de 2011.

BORWICK, John. Cfengine Introduction. **System Administrator Magazine**. 2006. online. Disponível em: <http://www.johnborwick.com/writing/cfengine.html>. Acesso em Maio de 2011.

BROCKMEIER, Joe "Zonker". Introduction to Puppet: Streamlined System Configuration. **Linux.com**. Júlio de 2010. Disponível em: <http://www.linux.com/learn/tutorials/325201-introduction-to-puppet-streamlined-system-configuration>. Acesso em: Maio de 2011.

CANONICAL Ltd. **About Ubuntu**: The Ubuntu story. 2011. online. Disponível em: <http://www.ubuntu.com/project/about-ubuntu>. Acesso em: Junho de 2011.

CFENGINE AS. **Use Cases**. 2011. online. Disponível em: <http://cfengine.com/pages/use_cases>. Acesso em: Junho de 2011.

CHRISTIAS, Panagiotis. **SSH(1)**: BSD General Commands Manual. 1994. online. Disponível em: <http://unixhelp.ed.ac.uk/CGI/man-cgi?ssh+1>. Acesso em: Junho de 2011.

CISCO SYSTEMS, Inc. **Cisco Network Virtualization Q&A**. 2006. online. Disponível em: <http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns431/ns658/net_qanda0900aecd804a16ae.pdf>. Acesso em: Junho de 2011.

CITRIX, Inc. **Citrix XenServer**: Product Overview. 2010. Disponível em: <http://www.citrix.com/%2Fsite%2Fresources%2Fdynamic%2Fsalesdocs%2FCitrix_XenServer_ProductOverview.pdf>. Acesso em: Maio de 2011.

DASMALCHI, Glenn. **Cloud Computing vs. Virtualization**. 2009. online. Disponível em: <http://www.youtube.com/watch?v=6pdL4aefX6Y&feature=player_embedded>. Acesso em: Maio de 2011. Vídeo.

DASTJERDI, Amir Vahid; TABATABAEI, Sayed Gholam Hassan; BUYYA, Rajkumar. An Effective Architecture for Automated Appliance Management System Applying Ontology-Based Cloud Discovery. In: 10th IEEE/ACM International Conference on

Cluster, Cloud and Grid Computing, 2010, Melbourne. **Anais**. Melbourne: IEEE/ACM, 2010, p. 104-112.

EUCALYPTUS SYSTENS, Inc. **Learn about cloud computing**. 2011. online. Disponível em: <http://open.eucalyptus.com/learn/what-is-eucalyptus>. Acesso em: Maio de 2011.

GARTNER, Inc. Gartner Says Cloud Computing Will Be As Influential As E-business. **Gartner**. 2008. online. Disponível em: <http://www.gartner.com/it/page.jsp?id=707508>. Acesso em: Junho de 2011.

KECSKEMETI, Gabor; TERSTYANSKY, Gabor; KACSUK, Peter; NEM, Zsolt. An approach for Virtual Appliance Distribution for Service Deployment. **Future Generation Computer Systems**, Amsterdam, v. 27, n. 3, março de 2011.

KREUTZ, Diego Luís. **Um Sistema de Gerenciamento Integrado e Automatizado de Serviços e Computadores de Redes Locais**. Santa Maria, 2005.

_____, Diego Luís. **FlexVAPS**: Um sistema de gerenciamento de virtual appliances para máquinas virtuais hererogêneas. Santa Maria, 2009.

KREUTZ, Diego L.; CHARÃO, Andrea. **FlexVAPs:** a system for managing virtual appliances in heterogeneous virtualized environments. In: 6th Latin American Network Operations and Management Symposium, 2009, Punta del Este.

LIMONCELLI, Thomas A.; HOGAN, Christina J.; CHALUP, Strata R. **The Practice of System and Network Administration**. 2. ed. Boston: Addison-Wesley, 2007.

LINUX MAGAZINE. **Google lança gerenciador GPL de servidores virtuais**. 2007. online. Disponível em: <http://www.linuxmagazine.com.br/ganeti>. Acesso em: Maio de 2011.

MARSHALL, David. Gartner: Only 25 percent of server workloads will be in a virtual machine by end 2010. **vmblog.com**. 2010. online. Disponível em: <http://vmblog.com/archive/2010/09/27/gartner-only-25-percent-of-server-workloads-will-be-in-a-virtual-machine-by-end-2010.aspx>. Acesso em: Junho de 2011.

McPHERSON, Amanda. Linux: The Operating System of the Cloud. **The Linux Foundation**. Disponível em: <http://www.linux.com/learn/whitepapers/doc/8/raw>. Acesso em: Maio de

2011.

MOERTEL, Tom. **A couple of tips for writing Puppet manifests**. Novembro de 2007. online. Disponível em: <http://blog.moertel.com/articles/2007/11/15/a-couple-of-tips-for-writing-puppet-manifests>. Acesso em: Maio de 2011.

NETCRAFT. **Most Reliable Hosting Company Sites in May 2011**. 2011. online. Disponível em: <http://uptime.netcraft.com/perf/reports/performance/Hosters?orderby=epercent&tn=may_2011>. Acesso: Junho de 2011.

NICCOLAI, James. Ballmer Still Searching for an Answer to Google. **PcWorld Business Center**. 2008. online. Disponível em: <http://www.pcworld.com/businesscenter/article/151568/ballmer_still_searching_for_an_answer_to_google.html>. Acesso em: Maio de 2011.

OPENNEBULA PROJECT LEADS. **About the OpenNebula.org Project**. 2011. online. Disponível em: <http://opennebula.org/about:about>. Acesso em: Junho de 2011.

OPENSTACK.ORG. **Building the Open Source Cloud**. online. Disponível em: <http://www.openstack.org/projects>. Acesso em: Junho de 2011.

_____. **Managing Virtual Machine Images**. 2011. online. Disponível em: <http://opennebula.org/documentation:rel2.2:img_guide>. Acesso em: Maio de 2011.

OPSCODE, Inc. **What is Chef**. 2011. online. Disponível em: <http://wiki.opscode.com/pages/viewpage.action?pageId=7274862>. Acesso em: Junho de 2011.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. **About**. 2011. online. Disponível em: <http://www.postgresql.org/>. Acesso em: Maio de 2011.

PUPPET, Labs. **Puppet Modules**. 2010. online. Disponível em: <http://forge.puppetlabs.com/>. Acesso em: Maio de 2011.

_____. **Featured Users**. 2011. online. Disponível em: <http://www.puppetlabs.com/customers/companies/>. Acesso em: Junho de 2011.

_____. **Learning**: Modules and Classes (Part One). 2011. online. Disponível em: <http://docs.puppetlabs.com/learning/modules1.html>. Acesso em: Junho de

2011.

RED HAT EMERGING TECHNOLOGIES. **Kernel Based Virtual Machine**. online. Disponível em: <http://www.linux-kvm.org/page/Main_Page>. Acesso em: Maio de 2011.

_____. **Libvirt**: The virtualization API. online. Disponível em: <http://libvirt.org/>. Acesso em: Maio de 2011.

_____. **Manage Virtual Machines**. online. Disponível em: <http://virt-manager.et.redhat.com/>. Acesso em: Maio de 2011.

RED HAT, Inc. **Red Hat Enterprise Virtualization for Servers 2.2**: Feature Matrix. 2010. Disponível em: <https://www.redhat.com/f/pdf/rhev/RHEV_DOC103_FeatureMatrix_1010_ch_web.pdf>. Acesso em: Maio de 2011.

_____. **The Evolution of Virtualization**: Qumranet joins Red Hat. 2010. online. Disponível em: <http://www.redhat.com/promo/qumranet/>. Acesso em: Maio de 2011.

_____. **Red Hat Enterprise Virtualization for Servers Demo**. online. Disponível em: <http://www.redhat.com/v/swf/rhev/demo.html>. Acesso em: Maio de 2011. Vídeo.

_____. **Creating a Virtual Machine Demo**. online. Disponível em: <https://www.redhat.com/v/swf/redhat_ss_createvm.html>. Acesso em: Maio de 2011. Vídeo.

SCHÄLING, Boris. **The Boost C++ Libraries**: Introduction. 2010. online. Disponível em: <http://en.highscore.de/cpp/boost/>. Acesso: Maio de 2011.

TOP500.Org. **Operating System share for 11/2010**. 2010. online. Disponível em: <http://www.top500.org/stats/list/37/osfam>. Acesso em: Junho de 2011.

VARGAS, Enrique. **High Availability Fundamentals** – Sun BluePrints. 2000. Disponível em: <http://www.sun.com/blueprints/1100/HAFund.pdf>. Acesso em: Novembro de 2010.

VENEZIA, Paul. Virtualization Shoot-out: Citrix, Microsoft, Red Hat, and VMware. **PCWolrd**. Abril de 2011. online. Disponível em:

<http://www.pcworld.com/businesscenter/article/225040-1/virtualization_shootout_citrix_microsoft_red_hat_and_vmware.html>. Acesso em: Maio de 2011.

VERAS, Manoel. **Virtualização de Servidores**. Rio de Janeiro: Escola Superior de Redes, 2011.

VMWARE, Ince. **Best Practices for Building Virtual Appliances**. 2007. Disponível em: <http://www.vmware.com/files/pdf/vam_best_practices_building.pdf?client=ubuntu&channel=cs&ie=UTF-8&q=virtual%20appliances%20best%20practices>. Acesso em: Novembro de 2010.

_____. **Virtual Appliances Demo**. online. Disponível em: <http://download3.vmware.com/media/vam/vam_demo.html>. Acesso em: Maio de 2011. Vídeo.

_____. **Disaster Recovery**. 2011. online. Disponível em: <http://www.vmware.com/solutions/continuity/disasterrecovery.html>. Acesso em:
Maio de 2011.

_____. **Learn about virtual appliances**. 2011. online. Disponível em: <http://www.vmware.com/appliances/getting-started/learn/overview.html>. Acesso
em: Maio de 2011.

_____. **Open Virtualization Format**: What is OVF?. 2011. online. Disponível em: <http://www.vmware.com/appliances/getting-started/learn/ovf.html>. Acesso em: Junho de 2011.

WILLIANS, Alex. The Virtualization Wars: Microsoft and Citrix v. VMware. **ReadWriteWeb**. Março de 2010. online. Disponível em: <http://www.readwriteweb.com/cloud/2010/03/the-virtualization-wars-micros.php>. Acesso em: Maio de 2011.

WIKINVEST. **VMware Inc. (VMW)**. online. Disponível em: <http://www.wikinvest.com/stock/VMware_Inc._(VMW)>. Acesso em: Maio de 2011.

Glossário

Apache2 – servidor HTTP *open source*

Cluster – conjunto de computadores operando de forma integrada (como um só computador) e partilhando seus recursos.

DHCP – é um protocolo para a configuração automática das informações de rede em comutadores

DRBD – é um sistema de armazenamento distribuído comumente utilizados em *clusters* de alta disponibilidade

Framework – plataforma de desenvolvimento que contém um conjunto de bibliotecas e classes

LVM – permite criar e gerenciar dispositivos de armazenamento de forma lógica e sob demanda

MySQL – é um sistema de gerenciamento de banco de dados relacional

Nuvem – refere-se, neste documento, a recursos computacionais disponibilizados em rede

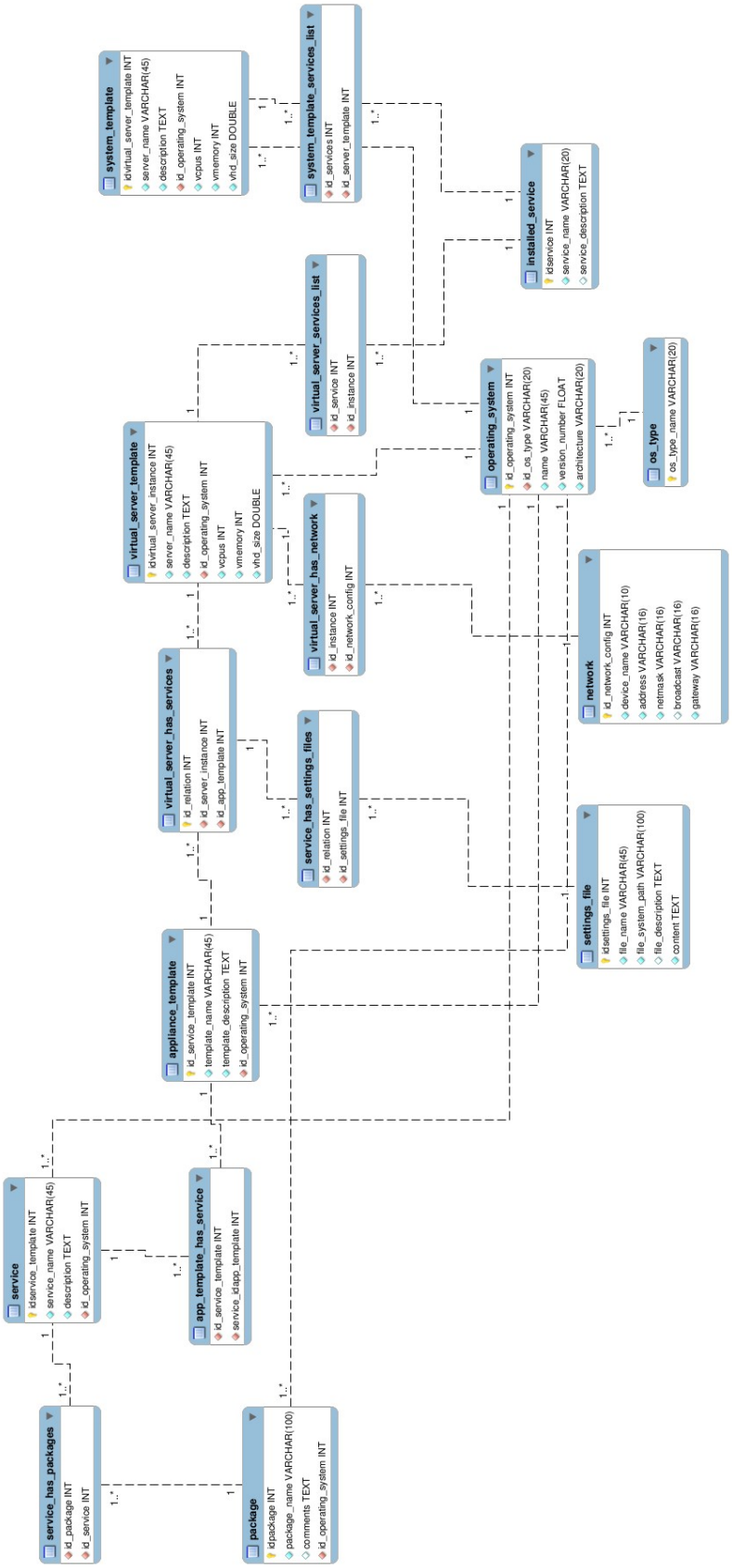
RAID-1 – RAID é são *arrays* de discos independentes redundantes, em sua configuração 1 os discos trabalham de forma “espelhada”, todos os discos recebem os dados identicamente

Storage – armazenamento através de dispositivos (discos)

Virtualização total (*full virtualization*) – é uma forma de virtualização na qual o *hypervisor* fornece ao sistema virtualização uma réplica do hardware subjacente. Sistemas virtualizados através desta forma de virtualização não requerem modificações no seu sistema pois tem a “ilusão” de estarem sendo executados sobre uma plataforma real

Wordpress – é uma aplicação para o gerenciamento de conteúdo para a publicação de sites e blogs

ANEXO 1 - Ilustração do banco de dados desenvolvido e utilizado para armazenamento de *templates*



ANEXO 2 - *Script* inicial.sql utilizado para a população do banco de dados

```
CREATE DATABASE v_servers;
\connect  v_servers;

-- -----------------------------------------------------
-- Table users
-- -----------------------------------------------------
CREATE TABLE users (
  username VARCHAR(45) PRIMARY KEY ,
  first_name VARCHAR(20) NOT NULL ,
  surname VARCHAR(100) NOT NULL ,
  password TEXT NOT NULL
);

INSERT INTO users (username, first_name, surname, password)
  VALUES ('admin', 'User', 'default', 'admin');

-- -----------------------------------------------------
-- Table os_type
-- -----------------------------------------------------
CREATE TABLE os_type (
  os_type_name VARCHAR(20) PRIMARY KEY
);

INSERT INTO os_type (os_type_name) VALUES ('LINUX');


-- -----------------------------------------------------
-- Table operating_system
-- -----------------------------------------------------
CREATE TABLE operating_system (
  id_operating_system SERIAL PRIMARY KEY ,
  id_os_type VARCHAR(20) NOT NULL ,
  name VARCHAR(45) NOT NULL DEFAULT 'Ubuntu' ,
  version_number REAL NOT NULL,
  architecture VARCHAR(45) NOT NULL DEFAULT 'x86_64' ,
  install_command VARCHAR(100) NOT NULL DEFAULT 'apt-get --yes install' ,
  FOREIGN KEY (id_os_type) REFERENCES os_type (os_type_name)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);


INSERT INTO operating_system (id_os_type, version_number) VALUES ('LINUX', '10.04');
INSERT INTO operating_system (id_os_type, version_number) VALUES ('LINUX', '10.10');
INSERT INTO operating_system (id_os_type, name, version_number) VALUES ('LINUX',
'Debian', '6.0');
```

```
-- ----------------------------------------------------
-- Table virtual_server_template
-- ----------------------------------------------------
CREATE TABLE virtual_server_template (
  idvirtual_server_template SERIAL PRIMARY KEY ,
  server_name VARCHAR(45) NOT NULL ,
  description TEXT NOT NULL ,
  author VARCHAR(45) NOT NULL ,
  date_time TIMESTAMP WITHOUT TIME ZONE NOT NULL DEFAULT now() ,
  id_operating_system INT NOT NULL ,
  vcpus INT NOT NULL DEFAULT 2 ,
  vmemory INT NOT NULL DEFAULT 512 ,
  vhd_size REAL NOT NULL DEFAULT 3.0,
  FOREIGN KEY (author ) REFERENCES users (username )
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  FOREIGN KEY (id_operating_system ) REFERENCES operating_system
(id_operating_system )
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

INSERT INTO virtual_server_template (server_name, description, author,
id_operating_system, vcpus)
  VALUES ('template-ubuntu-clean', 'Fresh install of Ubuntu 10.04.', 'admin', 1, 1);

-- ----------------------------------------------------
-- Table virtual_server_instance_template
-- ----------------------------------------------------
CREATE TABLE virtual_server_instance_template (
  idvirtual_server_instance SERIAL PRIMARY KEY ,
  server_name VARCHAR(45) NOT NULL ,
  description TEXT NOT NULL ,
  author VARCHAR(45) NOT NULL ,
  date_time TIMESTAMP WITHOUT TIME ZONE NOT NULL DEFAULT now() ,
  id_source_template INT NOT NULL ,
  id_operating_system INT NOT NULL ,
  vcpus INT NOT NULL DEFAULT 2 ,
  vmemory INT NOT NULL DEFAULT 512 ,
  vhd_size REAL NOT NULL DEFAULT 3.0,
  FOREIGN KEY (author ) REFERENCES users (username )
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  FOREIGN KEY (id_operating_system ) REFERENCES operating_system
(id_operating_system )
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  FOREIGN KEY (id_source_template )
    REFERENCES virtual_server_template (idvirtual_server_template )
    ON DELETE NO ACTION
```

```
    ON UPDATE CASCADE
);


-- -----------------------------------------------------
-- Table network
-- -----------------------------------------------------
CREATE TABLE network (
  id_network_config SERIAL PRIMARY KEY ,
  device_name VARCHAR(10) NOT NULL ,
  address VARCHAR(16) NULL ,
  netmask VARCHAR(16) NULL DEFAULT '255.255.255.0' ,
  broadcast VARCHAR(16) NULL ,
  gateway VARCHAR(16) NULL
);

INSERT INTO network (device_name,address,broadcast,gateway)
  VALUES ('eth0','192.168.122.199','192.168.122.255','192.168.122.1');


-- -----------------------------------------------------
-- Table server_template_has_network
-- -----------------------------------------------------
CREATE TABLE server_template_has_network (
  id_server_template INT NOT NULL ,
  id_network_config INT NOT NULL ,
  FOREIGN KEY (id_server_template) REFERENCES virtual_server_template
(idvirtual_server_template)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (id_network_config) REFERENCES network (id_network_config)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

INSERT INTO server_template_has_network (id_server_template, id_network_config)
VALUES (1, 1);


-- -----------------------------------------------------
-- Table instance_has_network
-- -----------------------------------------------------
CREATE TABLE instance_has_network (
  id_instance INT NOT NULL ,
  id_network_config INT NOT NULL ,
  FOREIGN KEY (id_instance) REFERENCES virtual_server_instance_template
(idvirtual_server_instance)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (id_network_config) REFERENCES network (id_network_config)
```

```
    ON DELETE CASCADE
    ON UPDATE CASCADE
);


-- -----------------------------------------------------
-- Table service
-- -----------------------------------------------------
CREATE TABLE service (
  idservice_template SERIAL PRIMARY KEY ,
  service_name VARCHAR(45) NOT NULL ,
  description TEXT NOT NULL ,
  author VARCHAR(45) NOT NULL ,
  date_time TIMESTAMP WITHOUT TIME ZONE NOT NULL DEFAULT now() ,
  id_operating_system INT NOT NULL ,

  FOREIGN KEY (author ) REFERENCES users (username )
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  FOREIGN KEY (id_operating_system ) REFERENCES operating_system
(id_operating_system )
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

-- DHCP
INSERT INTO service (service_name, description, author, id_operating_system)
  VALUES ('DHCP server', 'Standard DHCP server.
* By default listen to eth0', 'admin', 1);

-- Web server
INSERT INTO service (service_name, description, author, id_operating_system)
  VALUES ('Web server', 'Standard Web server.
* Default web server', 'admin', 1);

-- -----------------------------------------------------
-- Table package
-- -----------------------------------------------------
CREATE TABLE package (
  idpackage SERIAL PRIMARY KEY ,
  package_name VARCHAR(100) NOT NULL ,
  id_operating_system INT NOT NULL ,
  comments TEXT NULL ,
  FOREIGN KEY (id_operating_system ) REFERENCES operating_system
(id_operating_system )
    ON DELETE NO ACTION
    ON UPDATE CASCADE
);

-- DHCP
```

```
INSERT INTO package (package_name, id_operating_system, comments)
  VALUES ('dhcp3-server', 1, 'Install DHCP server for Ubuntu 10.04');

-- Apache2
INSERT INTO package (package_name, id_operating_system, comments)
  VALUES ('apache2', 1, 'Install Apache2 server for Ubuntu 10.04');

-- PHP
INSERT INTO package (package_name, id_operating_system, comments)
  VALUES ('php5 libapache2-mod-php5 php5-mysql', 1, 'Install PHP5 for Ubuntu 10.04');

-- MySQL
INSERT INTO package (package_name, id_operating_system, comments)
  VALUES ('mysql-server', 1, 'Install MySQL for Ubuntu 10.04');

-- -----------------------------------------------------
-- Table service_has_packages
-- -----------------------------------------------------
CREATE TABLE service_has_packages (
  id_package INT NOT NULL ,
  id_service INT NOT NULL ,
  FOREIGN KEY (id_package )
    REFERENCES package (idpackage )
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (id_service )
    REFERENCES service (idservice_template )
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

-- Service DHCP server has package dhcp3-server
INSERT INTO service_has_packages (id_package, id_service) VALUES (1, 1);

-- Service web server
INSERT INTO service_has_packages (id_package, id_service) VALUES (2, 2);
INSERT INTO service_has_packages (id_package, id_service) VALUES (3, 2);
INSERT INTO service_has_packages (id_package, id_service) VALUES (4, 2);



-- -----------------------------------------------------
-- Table settings_file
-- -----------------------------------------------------
CREATE TABLE settings_file (
  idsettings_file SERIAL PRIMARY KEY ,
  file_name VARCHAR(45) NOT NULL ,
  file_system_path VARCHAR(200) NOT NULL ,
  file_description TEXT NULL ,
  content TEXT NOT NULL ,
  author VARCHAR(45) NOT NULL ,
```

```
  FOREIGN KEY (author ) REFERENCES users (username )
    ON DELETE NO ACTION
    ON UPDATE CASCADE
);


-- ------------------------------------------------------
-- Table installed_service
-- ------------------------------------------------------
CREATE TABLE installed_service (
  idservice SERIAL PRIMARY KEY ,
  service_name VARCHAR(20) NOT NULL ,
  service_description TEXT NULL
);

INSERT INTO installed_service (service_name,service_description) VALUES ('DHCP','DHCP
protocol');
INSERT INTO installed_service (service_name,service_description) VALUES ('PHP','Script
programming language for web servers');
INSERT INTO installed_service (service_name,service_description) VALUES ('Apache2','Web
Server');
INSERT INTO installed_service (service_name,service_description) VALUES ('Drupal','CMS
tool');
INSERT INTO installed_service (service_name,service_description) VALUES
('Wordpress','CMS tool, blog');
INSERT INTO installed_service (service_name,service_description) VALUES
('SSH','Comunication between machines');
INSERT INTO installed_service (service_name,service_description) VALUES ('DNS','Domain
Name Service');
INSERT INTO installed_service (service_name,service_description) VALUES
('PostgreSQL','Database');
INSERT INTO installed_service (service_name,service_description) VALUES
('MySQL','Database');


-- ------------------------------------------------------
-- Table instance_services_list
-- ------------------------------------------------------
CREATE TABLE instance_services_list (
  id_service INT NOT NULL ,
  id_instance INT NOT NULL ,
  FOREIGN KEY (id_service )
    REFERENCES installed_service (idservice)
    ON DELETE NO ACTION
    ON UPDATE CASCADE ,
  FOREIGN KEY (id_instance )
    REFERENCES virtual_server_instance_template (idvirtual_server_instance )
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

```
-- ----------------------------------------------------
-- Table server_template_services_list
-- ----------------------------------------------------
CREATE TABLE server_template_services_list (
  id_services INT NOT NULL ,
  id_server_template INT NOT NULL ,
  FOREIGN KEY (id_services )
    REFERENCES installed_service (idservice )
    ON DELETE NO ACTION
    ON UPDATE CASCADE ,
  FOREIGN KEY (id_server_template )
    REFERENCES virtual_server_template (idvirtual_server_template )
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

INSERT INTO server_template_services_list (id_services,id_server_template) VALUES (6,1);

-- ----------------------------------------------------
-- Table scripts
-- ----------------------------------------------------
CREATE TABLE scripts (
  idscript SERIAL PRIMARY KEY ,
  script_name VARCHAR(100) NOT NULL ,
  content TEXT NOT NULL ,
  author VARCHAR(45) NOT NULL ,
  FOREIGN KEY (author )
    REFERENCES users (username )
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

-- DHCP
INSERT INTO scripts (script_name, content, author)
  VALUES ('DHCP restart', '/etc/init.d/dhcp3-server restart', 'admin');


-- ----------------------------------------------------
-- Table service_template
-- ----------------------------------------------------
CREATE TABLE service_template (
  id_service_template SERIAL PRIMARY KEY ,
  template_name VARCHAR(45) NOT NULL UNIQUE ,
  template_description TEXT NOT NULL ,
  author VARCHAR(45) NOT NULL ,
  date_time TIMESTAMP WITHOUT TIME ZONE NOT NULL DEFAULT now() ,
  id_operating_system INT NOT NULL ,
  idscript INT ,
```

```
  FOREIGN KEY (author )
    REFERENCES users (username )
    ON DELETE NO ACTION
    ON UPDATE CASCADE ,
  FOREIGN KEY (id_operating_system )
    REFERENCES operating_system (id_operating_system )
    ON DELETE CASCADE
    ON UPDATE CASCADE ,
  FOREIGN KEY (idscript )
    REFERENCES scripts (idscript )
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

-- DHCP
INSERT INTO service_template (template_name, template_description, author,
id_operating_system, idscript)
  VALUES ('DHCP server - simple', 'Standard DHCP server', 'admin', 1, 1);

-- Web Server
INSERT INTO service_template (template_name, template_description, author,
id_operating_system, idscript)
  VALUES ('Web server - simple', 'Standard Web server', 'admin', 1, 1);


-- -----------------------------------------------------
-- Table server_instance_has_services
-- -----------------------------------------------------
CREATE TABLE server_instance_has_services (
  id_relation SERIAL PRIMARY KEY ,
  id_server_instance INT NOT NULL ,
  id_service_template INT NOT NULL ,
  FOREIGN KEY (id_server_instance)
    REFERENCES virtual_server_instance_template (idvirtual_server_instance)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (id_service_template)
    REFERENCES service_template (id_service_template)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);


-- -----------------------------------------------------
-- Table service_has_settings_files -- relation
-- -----------------------------------------------------
CREATE TABLE service_has_settings_files (
  id_relation INT NOT NULL ,
  id_settings_file INT NOT NULL ,
  FOREIGN KEY (id_relation)
```

```
    REFERENCES server_instance_has_services (id_relation)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (id_settings_file)
    REFERENCES settings_file (idsettings_file)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);


-- -----------------------------------------------------
-- Table service_template_has_service
-- -----------------------------------------------------
CREATE TABLE service_template_has_service (
  id_service_template SERIAL PRIMARY KEY ,
  service_idservice_template INT NOT NULL ,
  FOREIGN KEY (id_service_template )
    REFERENCES service_template (id_service_template )
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (service_idservice_template )
    REFERENCES service (idservice_template )
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

-- DHCP
INSERT INTO service_template_has_service (id_service_template,
service_idservice_template)
  VALUES (1, 1);

-- Web Server
INSERT INTO service_template_has_service (id_service_template,
service_idservice_template)
  VALUES (2, 2);
```