Infraestrutura de Sítios Institucionais Utilizando Contêineres Docker

Carlos V. B. Santos[1], **Felipe E. dos Santos**[1], **Luiz C. B. Martins**[1]

[1] Centro de Informática – CPD
Universidade de Brasília (UnB) – Brasília, DF – Brazil

{carlosvbs, felipesantos,luizmartins}@unb.br

Resumo. *Este artigo descreve a migração da infraestrutura de sítios institucionais no qual utilizava-se jaulas (jails/chroot) para o uso dos contêineres através do Docker, criando uma nova infraestrutura com um nó central de requisições (proxy) junto com a implementação de vários mecanismos de segurança que não estavam habilitados. O nosso objetivo é ter uma infraestrutura simples de gerenciar e manter, possibilitando a expansão e a atualização, conforme a necessidade, sem interferir no eco-sistema e nos outros sítios.*

1. Introdução

Com a expansão das aplicações *Web* são inúmeros os serviços e aplicativos oferecidos por instituições públicas e privadas acessíveis através da Internet ou Intranet, substituindo os habituais *Softwares* que requeriam uma instalação em cada *Desktop*. Segundo o Ibope Media, em 2013 eramos 105 milhoes de internautas [Assencio 2013], sendo o Brasil o quinto país mais conectado na época. Hoje ocupamos a quarta posição com mais de 120 milhões, ultrapassando o Japão [1].

A grande quantidade de ambientes com diferentes configurações necessárias para disponibilização destes serviços *Web* traz grandes desafios para gerenciar estas plataformas no que tange tanto a configurações quanto a atualizações. Dentre estes serviços há a hospedagem de sítios institucionais e a dificuldade em gerenciar. A Universidade de Brasília (UnB) utiliza o *Content Management System* (CMS) Joomla! como padrão para publicação de sítios visto a sua facilidade para a publicação de informações, possibilitando assim que o gerenciamento de conteúdo pudesse ser descentralizado para as áreas fins [Martins 2017]. **Outro aspecto importante é a heterogeneidade dos sítios que estão em versão diferentes e utilizam muitas vezes componentes diversos que nem sempre aceitam essas atualizações**, além da maioria estarem simplesmente desatualizados. Desde a adoção do padrão Joomla! houve um grande aumento no número de sítios nos últimos anos que devido a arquitetura que visa a acoplamento de extensões e diversas versões utilizadas, no qual manter ambiente de hospedagem de sítios torna-se um desafio. O tema abordado então será a infraestrutura de sítios institucionais utilizando contêineres através da plataforma Docker, devido a sua agilidade, portabilidade e controle.

Atualmente na infraestrutura dos sítios institucionais, utilizam as jaulas (*jails*) com a ferramenta chroot para alterar o caminho padrão de diretórios e restringir o acesso dos sítios ao seu respectivo escopo. Entretanto, essa implementação gera uma problemática

[1] http://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2013/Individuals_Internet_2000-2012.xls

quanto a atualização do Sistema Operacional e suas bibliotecas que necessitam ser catalogadas e novamente direcionadas para cada jaula, demandando tempo e sem garantias de que as novas bibliotecas irão funcionar no ambiente.

Outro aspecto importante é a heterogeneidade dos sítios que estão em versão diferentes e utilizam muitas vezes componentes diferentes, os quais nem sempre aceitam estas atualizações, cuja a encontramm-se desatualizados. Ainda há a necessidade de atualização do Sistema Operacional que implica em muitos casos na falha operacional de vários sítios pois utilizam versões muito antigas de seus componentes, forçando o sistema a manter-se estagnado e comprometendo, assim a segurança.

Um contêiner é um aparato mínimo necessário para executar um determinado *software*, diferente de uma máquina virtual que agrega um sistema operacional inteiro. Com isso, somente as bibliotecas e as configurações essenciais para o funcionamento do *software* são necessários, tornando os contêineres uma opção eficiente, leve e auto-suficientes, garantindo que o sistema executado use sempre as mesmas configurações[Dua et al. 2014].

O Docker[2] é uma plataforma *open source* escrito em Go, cuja a uma linguagem de programação é desenvolvida no Google[3]. Essa plataforma não é um sistema de virtualização tradicional, pois no Docker nós temos recursos isolados os quais utilizam bibliotecas compartilhadas entre o host e o contêiner[Anderson 2015]. A figura 1 mostra como é estruturada a arquitetura Docker.

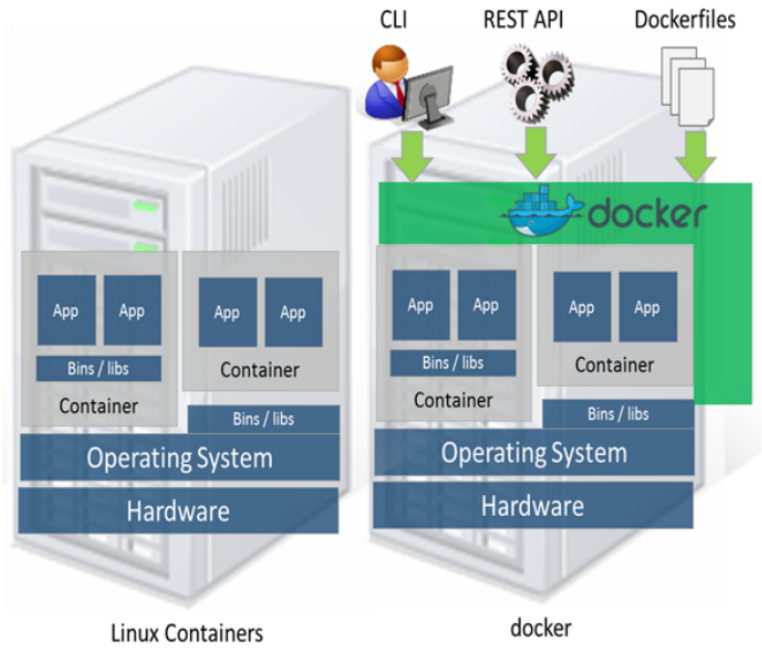

Figura 1. Arquitetura Docker

[2]https://www.docker.com/
[3]http://wwww.google.com

2. Metodologia

Visando solucionar esses problemas de infraestrutura, foi usado os contêineres como uma forma de isolar cada sítio no lugar das *jails* (*chroot*) e criou-se uma imagem Docker como referência, já com os recursos mais atualizados. Os sítios ficaram cada um em seu respectivo contêiner, dessa forma possibilitamos que caso algum deles necessite de bibliotecas adicionais para que elas possam ser disponibilizadas sem interferir no ecossistema, garantindo portanto, a independência e o isolamento de cada sítio. Além disso criou-se um contêiner para o Banco de Dados usando a imagem oficial do MariaDB[4] e efetuando uma ligação entre eles.

Foi criamos também um outro contêiner com Apache[5] e mod_proxy para receber todas as requisições e redirecioná-las para o *Virtual Host* adequado. Esse Proxy serve como ponto de convergência para a implementação do *Web Application Firewall* através do mod_security. Foi definido também um redirecionamento de URL(*Uniform Resource Locator*) da porta 80 para 443 a fim de forçar o uso de TLS/SSL, uma camada essencial para segurança, principalmente na necessidade de usar o painel de administração fornecendo usuário e senha. Outra camada de mitigação de ataques é o uso do Fail2Ban para efetuar o bloqueio através do Iptables dos endereços IP que usarem de força bruta para tentar ganhar autenticação ou ataques de negação de serviço. Todos esses itens abordados não estavam implementados na infraestrutura anterior.

Para a imagem Docker dos sítios foi utilizada como base a imagem Joomla! oficial disponível no sítio hub.docker.com que após os ajustes chamamos de site-php7. Para a imagem do *proxy* foi usada a imagem oficial do Debian em sua última versão (*latest*) e após os ajustes a chamamos de apache-proxy. Por fim a imagem do Banco de Dados não foi implementada alterações, mantendo a versão oficial do MariaDB (*latest*) também disponível no mesmo endereço. A figura 2 apresenta a arquitetura.

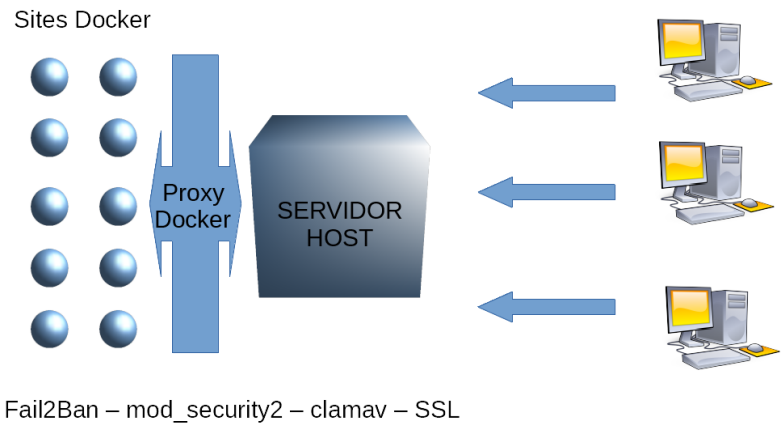

Figura 2. Arquitetura utilizando Contêineres e Docker

Foi fim, foram criados alguns arquivos de configuração que serviram de base ou template para todos os sítios implantados no contêineres conforme será descrito a seguir.

[4]https://mariadb.org/
[5]http://apache.org/

Diretório com o arquivo docker-compose.yml que conterá todas as configurações dos contêineres utilizados no ambiente:

```
/opt/docker/ambiente
  - docker-compose.yml
```

Diretório que servirá de volume para o contêiner de banco de dados:

```
/opt/docker/mysql
```

Diretório que conterá as chaves do certificado SSL:

```
/opt/docker/ssl
  - certs
  - private
```

Diretório com as configurações do proxy para os *Virtual Hosts*:

```
/opt/docker/apache2
  - logs
  - sites-enabled
    - siteXXX-proxy.conf
```

Diretório que conterá os sítios hospedados nos contêineres:

```
/opt/docker/sites
  - siteXXX
    - logs
    - htdocs
    - conf
      - siteXXX.conf
```

Com exceção do diretório ambiente, os demais servem de volumes para os contêineres e persistência dos dados no host além de armazenar as configurações necessárias para o correto funcionamento da infraestrutura.

3. Resultados

Conseguiu-se, então, definir uma infraestrutura segura e escalonável através da utilização dos contêineres. Obteve-se uma imagem com as configurações padronizadas e necessárias para o funcionamento dos sítios, no qual nos permite, se necessário, trata-los individualmente sem interferir nos demais. A figura 3 apresenta a diferença da arquitetura baseada em Máquina Virtual e a baseada em contêineres.

Uma outra possibilidade dos contêineres é o uso de cgroups, no qual permite especificar os limites de utilização dos recursos de *hardware* para cada contêiner, como memória, CPU, rede e I/O, uma capacidade que as jaulas, portanto, não permitiam, evitando assim, que um determinado processo possa exaurir os recursos do servidor em detrimento dos demais.

O uso dos contêineres nos permite também nos permite a facilidade de exportação do ambiente de homologação para a produção (*deploy*), de forma que todas as configurações definidas possam ser exportadas e importadas integralmente.

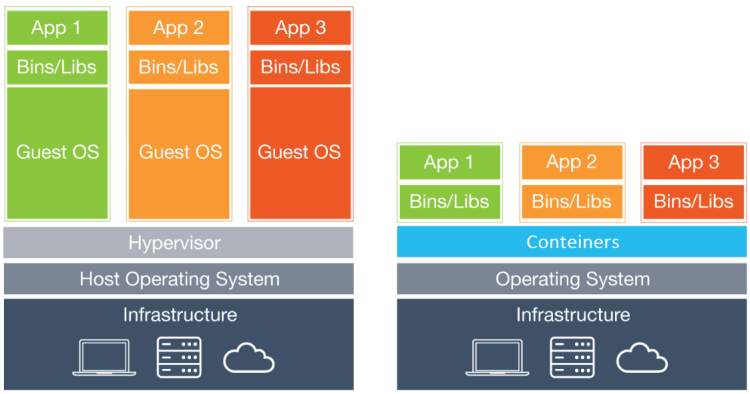

Figura 3. Virtual M vs Container

4. Conclusão

A utilização de contêineres proporcionou um ambiente mais seguro, através da mitigação da superfície de ataque com atualizações de *softwares*, implementação de vários aspectos de segurança que não existiam no ambiente antigo, como o TLS/SSL, Fail2Ban, mod_security e iptables.

Obtivemos o resultado que esperavamos com as alterações implementadas na nossa infraestrutura o qual nos permite gerenciar e implementar especificidades para cada contêiner, sem interferir nos demais. Além disso, a convergência das requisições para um nó central, possibilita a criação e a gerência de regras de segurança que são aplicadas a todos, não necessitando consequentemente de configuração desses mecanismos individualmente, e assim ganhar tempo comparada ao processo de enjaulamento utilizado anteriormente. Outro fator importante dessa trabalho é a possibilidade de aplicar essa infraestrutura em outros ambientes, como por exemplo, nos serviços em Java que estão atualmente em uso sendo este ponto a ser tratato futuramente.

Referências

Anderson, C. (2015). Docker [software engineering]. *IEEE Software*, 32(3):102–c3.

Assencio, C. (2013). Ibope aponta que acesso à internet cresce 3 `http://exame.abril.com.br/tecnologia/ibope-aponta-que-acesso-a-internet-cresce-3-no-2o-tri/`. Acessado: 2017-03-27.

Dua, R., Raja, A. R., and Kakadia, D. (2014). Virtualization vs containerization to support paas. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 610–614. IEEE.

Martins, L. C. B. (2017). Despacho cpd ssi sites 0683276. Technical report, Centro de Informática - UnB.