Automação de instalação e gerenciamento de servidores utilizando FAI e SaltStack

Eduardo A. Speroni[1], **Luciano A. Cassol**[1], **Maurício da V. de Almeida**[1]

[1]Centro de Processamento de Dados – Universidade Federal de Santa Maria (UFSM)
Av. Roraima nº 1000 – 97105-900 – Santa Maria – RS – Brasil

{eduardosperoni,lcassol}@ufsm.br, malmeida@inf.ufsm.br

Resumo. *A gerência de configuração de software é uma abordagem disciplinada que permite que o software evolua controladamente. Quando não automatizada, este é um processo que demanda demasiado tempo e cuidado, sendo altamente suscetível a erros. Este trabalho apresenta uma abordagem para a automatização deste processo, desde apertar o botão de ligar da máquina até o funcionamento pleno de seus serviços. Utilizando o método apresentado, foi possível gerenciar máquinas Linux e Windows, com o processo completo não levando mais de 15 minutos.*

1. Introdução

A gerência de configuração de software (GCS) é considerada o controle da evolução de sistemas complexos [Estublier 2000]. Especificamente, a GCS é uma abordagem disciplina que permite que o software evolua controladamente, contribuindo para satisfazer restrições de qualidade e de cronograma.

Este artigo apresenta a implantação de uma gerência de configuração através das ferramentas *Fully Automatic Installation* (FAI) e *SaltStack* na Universidade Federal de Santa Maria (UFSM), padronizando o processo de instalação e configuração de software em máquinas virtuais e físicas. Esta abordagem visa automatizar o processo desde a inicialização da máquina até o funcionamento pleno de seus serviços.

A organização deste artigo está dividida da seguinte maneira: na próxima seção são tratados os métodos utilizados para a implantação, em seguida são apresentados os resultados obtidos, e finalmente são apresentadas as considerações finais.

2. Métodos

Para uma completa automação da configuração de servidores, foram utilizadas duas ferramentas: *Fully Automatic Installation* (FAI) para a instalação do sistema base na máquina física ou virtual, e *SaltStack* para a instalação e configuração dos softwares e serviços que a máquina executará.

Conforme mostra a Figura 1, o FAI instala um sistema Linux em máquinas físicas ou virtuais, enquanto o Salt faz a configuração de sistemas Linux e Windows. No caso da imagem, o Salt configurou libvirt e openssh no servidor de virtualização, mysql e openssh na VM Linux e os serviços de componente (COM+) na VM Windows.

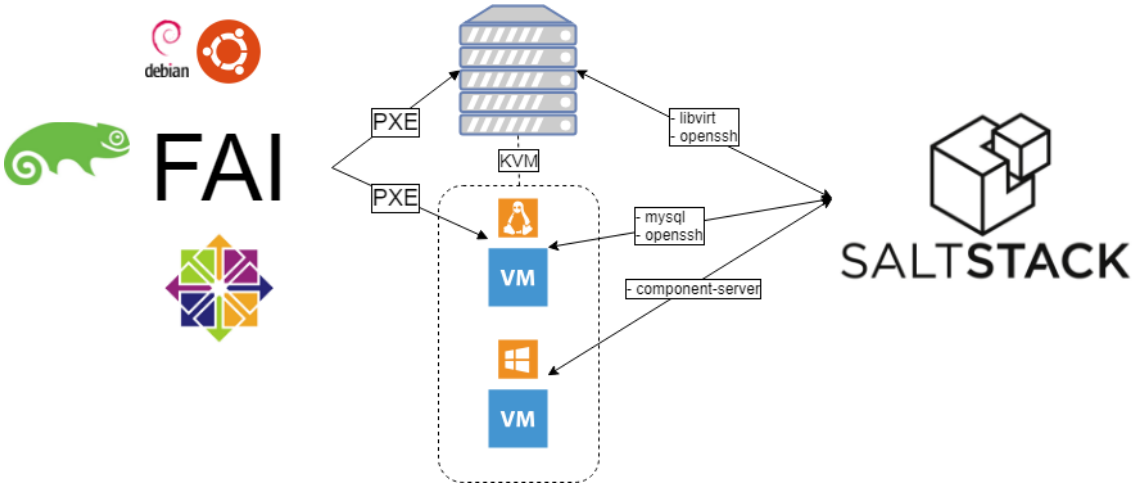

Figura 1. Diagrama do método utilizado

2.1. Fully Automatic Installation

O *Fully Automatic Installation* (FAI) é uma ferramenta não-interativa criada pra instalar, customizar e administrar sistemas Linux em computadores, máquinas virtuais e ambientes chroot [FAI 2018]. O uso desta ferramenta possibilita a automação da instalação do sistema com os pacotes básicos para seu funcionamento.

Durante o processo de instalação do FAI, é criado um *nfsroot* contendo o sistema base a ser instalado. Este diretório é montado na máquina destino via *Network File System* (NFS) durante o processo de boot via rede utilizando o *Preboot Execution Environment* (PXE) [FAI 2018].

O sistema utiliza o conceito de classes que determinam quais configurações a máquina deve ter. Se as classes `DEBIAN BRAZILIAN SALT` são definidas, a máquina instalará o Debian contido no *nfsroot*, configurará a linguagem PT-BR e instalará o SaltStack. Estas configurações estão divididas nos seguintes diretórios:

- **basefiles:** arquivos .tar.xz que serão extraídos para a máquina destino, em vez de utilizar o sistema do *nfsroot*, o que permite a instalação de outras distribuições Linux. Exemplo: XENIAL64.tar.xz (Ubuntu 16.04)
- **class:** composta de scripts que definem as classes da máquina, assim como arquvos `.var` que definem variáveis de ambiente. Exemplo: DEBIAN.var
- **debconf:** configurações iniciais dos pacotes a serem instalados.
- **disk_config:** partições, opções de montagem e sistemas de arquivos de cada disco. Exemplo: arquivo DEBIAN criando uma partição `/boot` de 150MB e com a partição `/` ocupando o resto do disco (no mínimo 10GB):

```
1 disk_config disk1 disklabel:msdos bootable:1 fstabkey:uuid
2 primary /boot 150M ext4 rw,noatime createopts="-L
    BOOT"
3 primary / 10G- ext4 rw,noatime,errors=remount-ro
    createopts="-L ROOT"
```

- **files:** arquivos que devem ser criados ou substituídos durante a instalação: Exemplo: `files/etc/ssh/sshd_config/DEBIAN` insere o conteúdo do arquivo DEBIAN em `files/etc/ssh/sshd_config`.

- **hooks:** scripts a ser executado em determinada parte do processo de instalação. Exemplo: `debconf.CENTOS` será chamado no passo de debconf quando conter a classe `CENTOS`
- **package_config:** pacotes que devem estar presentes ou ausentes
- **scripts:** diretório com scripts que devem ser executados em ordem alfabética. Exemplo: `scripts/SALT/10-instala.sh` será executado quando a classe `SALT` estiver presente.
- **tests:** scripts com testes para verificar se a instalação completou com sucesso.

Após definidas as classes e configurações iniciais, basta cadastrar o IP e hostname da máquina no arquivo `/etc/hosts`, adicionar no DHCP para que seja realizado o boot PXE, e executar o comando `fai-chboot` para marcar a máquina para a instalação. Desta maneira, é possível instalar qualquer distribuição Linux, utilizando configurações diferentes para máquinas virtuais e físicas, assim como instalar o SaltStack para que possam ser configuradas posteriormente.

2.2. SaltStack

O SaltStack é uma ferramenta de gerência de configuração de software altamente escalável que pode ser usado para orquestrar, executar comandos remotos em qualquer infraestrutura e configurar qualquer tipo de *stack* de aplicação [Sal 2018]. Concorrendo contra gigantes como Puppet e Chef, o Salt destaca-se por sua facilidade de instalação, configuração e desenvolvimento, e é utilizado em diversas empresas, como Cloudflare, Hulu, LinkedIn e Lyft.

Utilizando uma arquitetura cliente-servidor, o servidor é chamado de *master*, e os clientes de *minions*. A ferramenta pode ser divida em duas partes essenciais:

- **state** é o "estado" da máquina, ou seja, a sua configuração. Os estados são sempre visíveis por todos os *minions*, o que reforça que eles não devem ser utilizado para guardar informações sigilosas. Seu diretório normalmente fica em `/srv/salt/`.
- **pillar** é o chamado "pilar de sal", nele estão todas as "variáveis" que serão utilizadas na configuração. Quando um estado é executado, o *master* renderiza este *pillar* e entrega ao *minion* apenas os dados que ele pode ter acesso. Seu diretório normalmente fica em `/srv/pillar/`.

Os arquivos `.sls` (*SaLt State files*) são primariamente *templates* Jinja2 [Jin 2018] que são renderizado para o formato YAML. Nos diretórios raízes de *states* e *pillars*, um arquivo `top.sls` contém as informações de quais são os estados e pilares de cada *minion* [Sal 2018]. No Código 1, o arquivo `top.sls` faz com que o Salt carregue os arquivos `mysql/init.sls` (ou `mysql.sls`) e `mysql/python.sls`.

Código 1. Exemplo de top.sls

```
1 base:
2   'minion01':
3     - mysql
4     - mysql.python
```

Um conjunto de estados é chamado de fórmula. Diversas fórmulas são distribuídas pela equipe de desenvolvimento do SaltStack ou pela comunidade. A fórmula *mysql-formula*, por exemplo, é composta dos estados:

- **mysql:** Meta-estado que instala e configura o servidor mysql.
- **mysql.client:** Instala o cliente mysql.
- **mysql.server:** Instala o servidor mysql e inicia o serviço.
- **mysql.disabled:** Mantém o serviço desabilitado.
- **mysql.database:** Cria e gerencia os bancos de dados.
- **mysql.python:** Instala as dependências para utilizar o mysql em python.
- **mysql.user:** Cria e gerencia os usuário, assim como seus GRANTs.
- **mysql.dev:** Instala as bibliotecas de desenvolvimento
- **mysql.repo:** Utiliza o repositório oficial para instalar os pacotes.

Desta maneira, para a instalação e configuração de um servidor mysql, com um usuário e banco de dados chamados "zabbix" e senha "senhateste", com as bibliotecas de *Python* instaladas, basta utilizar o arquivo descrito no Código 1 e o *Pillar* do Código 2.

Código 2. Pillar de configuração do mysql

```
1  mysql:
2    server:
3      root_password: 'rootpw'
4    database:
5      - zabbix
6    user:
7      zabbix:
8        password: 'senhateste'
9        databases:
10         - database: zabbix
11           grants: ['all privileges']
```

Para reforçar a segurança, é possível criptografar qualquer configuração utilizando GPG. Esta abordagem permite que dados sigilosos do *pillar* possam ficar expostas em repositório *git* para que seja feito o versionamento da configuração.

3. Resultados

Utilizando o FAI (Seção 2.1) e o Salt (Seção 2.2), todas máquinas criadas ou reconfiguradas a partir de 2016 estão no novo sistema de configuração padronizada. Ao executar um `state.apply` em todos os 56 *minions*, o processo demorou 6 minutos para verificar que todas as configurações estavam no estado requisitado.

São utilizadas apenas duas classes diferentes para o FAI: uma para máquinas virtualizadoras e outra para provedoras de serviços, o que simplifica o processo, visto que o restante das configurações são realizadas pelo Salt. O processo completo de instalação de uma nova máquina dá-se da seguinte forma:

```
1 ./adiciona_dhcp.sh ip mac hostname
2 ./adiciona_chboot.sh ip
3 sleep 300 # aguarda 5 minutos até concluir a instalação
4 salt-key -a hostname # aceita a chave pública do minion, habilitando
      a configuração
5 salt hostname state.apply # aplica a configuração definida
```

Alguns minutos após a máquina ser ligada, ela conclui a sua instalação e registra-se no Salt para que seja configurada. Consideram-se os Códigos

3 e 4 como os arquivos `top.sls` dos estados e pilares, respectivamente.

Código 3. state top.sls

```
base:
  '* and not G@kernel:Windows':
    - match: compound
    - openssh
    - openssh.config
    - ntp.ng
    - locale
    - root-password
```

Código 4. pillar top.sls

```
base:
  '*':
    - selinux_padrao
    - ssh_default
    - ntp_default
    - zabbix_agent
    - locale-default
    - firewall.nostrict
```

Após configurada a máquina, basta utilizar os estados já existentes e novos pilares para configurar um novo serviço, ou pilares já existentes para obter um serviço igual à outro configurado anteriormente. Cerca de 60 fórmulas estão em uso atualmente, além de cerca de 40 arquivos de estado individuais, sejam auxiliares ou para tarefas simples, configurados por um total de 91 *pillars*.

Devido à velocidade e praticidade de desenvolvimento, foi possível gerenciar serviços COM+ (SIE) em máquinas Windows com *deploy* automático de 50 componentes com apenas 606 linhas de código em Python, processo que é realizado manualmente em cerca de 10 servidores. A execução deste componente leva em torno de 15 segundos para não detectar mudanças em uma máquina já configurada e de 5 a 10 minutos para configurar uma máquina recém criada.

4. Conclusão

Utilizando as ferramentas descritas neste trabalho, foi possível centralizar todo o gerenciamento do parque de máquinas em um só local com controle de versões para as configurações. A facilidade de desenvolvimento e uso permitem a replicação de qualquer máquina gerenciada, inclusive do *master* com mínimo retrabalho.

A solução provou-se eficaz para a necessidade da instituição. A automação da instalação de máquinas propiciou uma maior velocidade no processo e uma redução da carga de trabalho em geral do setor. A automação de *deploy* dos servidores SIE, que está em fase final de teste, poupará horas semanais de trabalho que é realizado por 2 funcionários, além das horas gastas somente na configuração de um novo servidor SIE.

References

(2018). Fai guide (fully automatic installation). `https://fai-project.org/fai-guide/`. Acessado em 23/04/2018.

(2018). Jinja2 (the python template engine). `http://jinja.pocoo.org/`. Acessado em 23/04/2018.

(2018). Saltstack documentation. `https://docs.saltstack.com/en/latest/`. Acessado em 23/04/2018.

Estublier, J. (2000). Software configuration management: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 279–289. ACM.