SaltStack 基础

你也许还没有听说过 Salt Stack 和 Chef 跟他们差不多，但是 Salt 是用 Python 写的，且对于设备的要求是相当轻量级的，使用起来相当容易(在我看来)，Salt 通讯层使用 0MQ（http://www.zeromq.org ），这是使得它很快速。并且它是完全开源的，遵守 Apche2 开源协议.拥有一个活跃和高效的开源社区。

Salt 和 Puppet Chef 一样可以让你同时在多台服务器上执行命令也包括安装和配置软件。Salt 有两个主要的功能：配置管理和远程执行。

Salt 包含两个工具，第一个是远程执行，虽然没有一个清晰的描述，但是假如你想要一个配置管理和远程执行的工具，在 Salt 中你可以找到很多方法。这可以让你登陆一台主服务器然后同时执行命令在一台或者多台服务器上。使用 Salt，你仅仅需要在主服务器上输入命令，它会在每台机器上甚至一个服务器群组执行。

第二，Salt 能够存储配置指令，然后引导其他机器按照这些指令作出动作，如，安装软件，更改软件的配置，反馈这个任务成功执行与否。

你可以轻易地使用 Salt 安装一个额外的包，并且仅仅通过一个命令配置这个包。

Salt 是不断变更的，也许当你读到本文时，有些事情已经改变，你可以在这里找到最新的文档：http://docs.saltstack.org/en/latest/index.html

SaltStack 是什么

Salt 是一种和以往不同的基础设施管理方法，它是建立在大规模系统高速通讯能力可以大幅提升的想法上。这种方法使得 Salt 成为一个强大的能够解决基础设施中许多特定问题的多任务系统。远程执行引擎是 Salt 的核心，它能够为多组系统创建高速、安全的双向通讯网络。基于这个通许系统，Salt 提供了一个非常快速、灵活并且容易使用的配置管理系统，称之为“Salt States”。

假如你有百十来台服务器，你应该怎么做？想象一下你每次一台一台的登陆这些服务器，执行同样的命令在这些 100 台的服务器中并且编辑同一个配置文件，你可以想象吗？这是多么的苦逼！仅仅更新一下用户的密码策划就要用掉数天的时间，并且你可能出现错误的操作，怎么样通过一个命令一次完成所有服务器的操作？怎么解决？答案就是， Salt Stack！

SaltStack 安装

各平台的安装参考官网：http://docs.saltstack.com/en/latest/topics/installation/index.html

如果采用 yum 源安装，可以安装 epel 源

```
rpm -Uvh epel-release-X-Y.rpm
On the salt-master, run this:
yum install salt-master
On each salt-minion, run this:
yum install salt-minion
```

这里 yum 会默认地安装依赖包

用 epel 源安装时可能会出现无法连接上的情况，可以挂个 vpn 试试，或者是源码安装

Git 源码:

```
git clone https://github.com/saltstack/salt.git
```

依赖包:

Salt should run on any Unix-like platform so long as the dependencies are met.

- Python 2.6 >= 2.6 <3.0
- msgpack-python - High-performance message interchange format
- YAML - Python YAML bindings
- Jinja2 - parsing Salt States (configurable in the master settings)
- MarkupSafe - Implements a XML/HTML/XHTML Markup safe string for Python
- apache-libcloud - Python lib for interacting with many of the popular cloud service providers using a unified API
- Requests - HTTP library

Depending on the chosen Salt transport, ZeroMQ or RAET, dependencies vary:

- ZeroMQ:
  - ZeroMQ >= 3.2.0
  - pyzmq >= 2.2.0 - ZeroMQ Python bindings
  - PyCrypto - The Python cryptography toolkit
  - M2Crypto - "Me Too Crypto" - Python OpenSSL wrapper
- RAET:
  - libnacl - Python bindings to libsodium
  - ioflo - The flo programming interface raet and salt-raet is built on
  - RAET - The worlds most awesome UDP protocol

可选依赖

mako - 一个可选的 Salt States 解析器(在 master 配置文件中配置)

gcc - 动态 Cython 模块编译器

依赖包自行安装

Salt defaults to the ZeroMQ transport, and the choice can be made at install time, for example:

```
python setup.py install --salt-transport=raet
```

This way, only the required dependencies are pulled by the setup script if need be.

If installing using pip, the `--salt-transport` install option can be provided like:

```
pip install --install-option="--salt-transport=raet" salt
```

升级 SALT

当升级 Salt 的时候，master(s)应该首先升级。masters 向后兼容，minions 运行版本比 masters 版本新是不能保证运行正常的。

只要可能，新 masters 向后兼容旧 minions 的特性就会被保留下来。一般来说，唯一例外的

情况是有安全漏洞的情况下。

SaltStack 配置与测试

master 端：

```
# service salt-master start
```

client 端：

```
# sed -i 's/#master: salt/master: master/' /etc/salt/minion
# service salt-minion start
```

在 master 端接收 client 端 key：

```
#salt-key -L
Accepted Keys:
client
Unaccepted Keys:
Rejected Keys:
# salt-key –A
```

测试

```
#salt '*' test.ping
client:
    True
#salt client cmd.run uptime
client:
  13:25:24 up  2:29,  1 user,  load average: 0.00, 0.00, 0.00
```

SaltStack 常用模块

更多 salt 模块访问：http://docs.saltstack.com/en/latest/ref/modules/all/index.html

Salt 拥有一个巨大的函数库可用于执行，而且 Salt 函数是自带文档说明的。在 minions 上执行 **sys.doc** 函数可以查看哪些函数可用：

```
salt '*' sys.doc
```

这会显示一个非常大的可用函数和函数文档列表。

cmd.run

文档`cmd </ref/modules/all/salt.modules.cmdmod>`模块包含在 minions 上执行 shell 命令的函数，比如模块`cmd.run <salt.modules.cmdmod.run>`和模块`cmd.run_all <salt.modules.cmdmod.run_all>`：

```
salt '*' cmd.run 'ls -l /etc'

salt client cmd.run "ping -c 2 baidu.com"
```

pkg.install

pkg 函数会自动将本地系统包管理器映射到相同的 salt 函数。这意味着 pkg.install 在基于 Red Hat 系统上将使用 yum 而在 Debian 系统上则使用 apt 来安装包，等等。

```
salt '*' pkg.install httpd
```

模块函数`network.interfaces <salt.modules.network.interfaces>`将会列出 minion 上的所有接口，以及它们的 IP 地址，子网掩码，MAC 地址等：

```
salt '*' network.interfaces
```

更改输出模式

```
salt myminion grains.item pythonpath --out=pprint

salt client user.info root --out=pprint
{'client': {'fullname': 'root',
            'gid': 0,
            'groups': ['root'],
            'home': '/root',
            'homephone': '',
            'name': 'root',
            'passwd': 'x',
            'roomnumber': '',
            'shell': '/bin/bash',
            'uid': 0,
            'workphone': ''}}
```

SaltStack 配置管理

一个安装 vim 的例子：

创建基础配置/TOP 文件

```
#cat top.sls
base:
 '*':
   - vim
```

base 语法告诉 Salt 这是基础配置文件，可以应用在所有机器上。通配符'*'的目标是所有机器。'- servers'指令可以是任意值，运行你识别哪些质量可以使用。再选择一些其它有用的配置。这个条目还指向一个特别的配置，用于安装 libpam-cracklib。

创建特定的服务器的配置文件

保存 top.sls 后，在/srv/salt 目录下创建 vim.sls 文件。这个文件包含特定的配置，包括安装包的名称，也可指向另外的配置文件。在 vim.sls 中，加入如下行：

```
#cat vim.sls
vim-enhanced:
  pkg:
    - installed
```

第一行是包管理工具可识别的软件包名称。以 Apache HTTP server 为例，在基于 apatitude 的包管理中叫做 apache2，而在基于 yum 的包管理中叫做 httpd。确保针对包管理工具使用正确的名字。也可以使用 Salt 的 grains 进行包管理。查看参考文档以获得更多信息，以及在 SLS 文件使用 grains 的例子 (http://salt.readthedocs.org/en/latest/topics/tutorials/states_pt3.html#using-grains-in-sls-modules).

第 2 和第 3 行告诉 Salt 如何处理这个包——本例是安装这个包。要删除一个包，你只需要修改'- installed'为'- removed'即可。记住，空格很重要！第二行**'pkg:'**前有两个空格，第三行**'- installed'**前有四个空格。如果遇到任何错误，请通过在线 **YAML** 解析器检查语法。

创建特定包的配置文件：

Salt 作为一个安全文件服务器，并把文件拷贝到远程的从服务器上。在 servers.sls 增加如下行：

```
#cat vim.sls
vim-enhanced:
  pkg:
    - installed
```

```
/etc/vimrc:
 file:
  - managed
  - source: salt://vim/vimrc
  - require:
    - pkg: vim-enhanced
```

注意第四行，它告诉 Salt 你的特殊文件的位置，这一行后面的行，即第 5 行，告诉 Salt 这个包的依赖包。
这行 salt://映射到你的主服务器/srv/salt 目录。
保存了 vim.sls 文件后，在/srv/salt 目录下创建 vim 目录。这里用来存储 vim 包的配置文件。

当你在安装软件和配置文件的时，有时候你想要在测试服务器上先行安装，然后以你的需要配置。你可以拷贝配置文件到/srv/salt 目录，这样子，你可以在部署到生成环境之前测试他们。
现在你的配置测试通过，现在你可以把配置文件随着安装 vim 分发到每台从机器上了。/srv/salt 目录应该如下：

```
/srv/salt
   top.sls
   vim.sls
   /vim
     vimrc
```

这里我把 vim 的配置作为一个例子，所有的其他软件配置都和这样差不多。例如，你可以比较容易的通过修改 httpd.conf 来实现虚拟主机和主机头的配置。
所有的 sls 文件准备好以后，最后一步是告诉 Salt 配置远程机器。state.highstate 是触发这些同步的命令。使用先前的语法格式，目标位所有机器，键入以下格式的命令：

```
salt '*' state.highstate
```

片刻后，从服务会反馈像如下成功的信息：

```
#salt client state.highstate
client:
        ID: vim-enhanced
   Function: pkg.installed
     Result: True
    Comment: The following packages were installed/updated: vim-enhanced.
    Started: 16:26:03.324350
   Duration: 33265.113 ms
    Changes:
             ----------
             vim-enhanced:
```

```
              ----------
              new:
                  7.2.411-1.8.el6
              old:
        ID: /etc/vimrc
  Function: file.managed
    Result: True
   Comment: File /etc/vimrc updated
   Started: 16:26:36.590170
  Duration: 12.127 ms
   Changes:
            ----------
            diff:
                ---
                +++
                @@ -62,3 +62,42 @@
                 " Don't wake up system with blinking cursor:
                 " http://www.linuxpowertop.org/known.php
                 let &guicursor = &guicursor . ",a:blinkon0"
                +
                +
                +
                +"add by jimmy"
                +function AddTitle()
……省略部分内容……
Summary
------------
Succeeded: 2 (changed=2)
Failed:    0
------------
Total states run:     2
```

再来一个例子（安装 **httpd**）

```
#ls
httpd  httpd.sls  top.sls  vim  vim.sls

[ root@My-CDN-MacHine /srv/salt]
#cat top.sls
base:
  '*':
    - vim
    - httpd
```

```
[ root@My-CDN-MacHine /srv/salt]
#cat httpd.sls
httpd:
  pkg:
    - installed
/etc/httpd/conf/httpd.conf:
  file:
    - managed
    - source: salt://httpd/httpd.conf
    - require:
      - pkg: httpd

[ root@My-CDN-MacHine /srv/salt]
#ls httpd
httpd.conf

[ root@My-CDN-MacHine /srv/salt]
#ls -R
.:
httpd  httpd.sls  top.sls  vim  vim.sls

./httpd:
httpd.conf

./vim:
vimrc
```

来看看执行结果：

```
#salt client state.highstate
client:
----------
          ID: vim-enhanced
    Function: pkg.installed
      Result: True
     Comment: Package vim-enhanced is already installed.
     Started: 16:59:07.372786
    Duration: 1565.873 ms
     Changes:
----------
          ID: /etc/vimrc
```

```
    Function: file.managed
      Result: True
     Comment: File /etc/vimrc is in the correct state
     Started: 16:59:08.939496
    Duration: 4.13 ms
     Changes:
----------
          ID: httpd
    Function: pkg.installed
      Result: True
     Comment: Package httpd is already installed.
     Started: 16:59:08.943783
    Duration: 0.57 ms
     Changes:
----------
          ID: /etc/httpd/conf/httpd.conf
    Function: file.managed
      Result: True
     Comment: File /etc/httpd/conf/httpd.conf updated
     Started: 16:59:08.944550
    Duration: 13.035 ms
     Changes:
              ----------
              diff:
                  ---
                  +++
                  @@ -133,7 +133,7 @@
                   # prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
                   #
                   #Listen 12.34.56.78:80
                  -Listen 80
                  +Listen 9999

                   #
                   # Dynamic Shared Object (DSO) Support
                  @@ -1007,3 +1007,10 @@
                   #    ErrorLog logs/dummy-host.example.com-error_log
                   #    CustomLog logs/dummy-host.example.com-access_log common
                   #</VirtualHost>
                  +Alias /sys_init/ "/home/sys_init/"
                  +<Directory "/home/sys_init/">
                  +Options Indexes FollowSymLinks
```

```
            +AllowOverride None
            +Order allow,deny
            +Allow from all
            +</Directory>

Summary
------------
Succeeded: 4 (changed=1)
Failed:    0
------------
Total states run:     4
```

本例中只有一台从服务器，相对来说是一个简单的例子，但是想象一下，假如你使用 Salt 配置管理安装了 LAMP web 服务器，你省了多长时间呀。把这些配置放在文本文件中，使得你快速高效的创建同样的服务器。

总结得挺不错的：

现在你有能力一次在很多台机器上执行远程命令并且能够把配置文件存储在易维护的文本中。也可以安装特殊的软件包。

刚开始的时候，多花点心思。您可以根据自己的特定配置创建一个或多个服务器，并且下载不同的包到每台机器上。Salt 也可以不按顺序执行，有些命令会同时执行，假如其中一台执行失败，其他依然不受影响继续执行。

虽然在安装 Salt 比较费时，但是你以后会得到极大的好处，特别是可以让你创建特定的服务器和可重复使用的配置。

访问 Salt 项目得到更多的细节，多关注邮件列表和用户文档以及一些例子。你会发现社区会非常热心的帮助你处理问题。

Salt states 深入研究

很多最强大、最有用的工程解决方案都是基于简单原则建立起来的。Salt States 也竭尽全力做到那样：K.I.S.S.(Keep It Stupidly Simple 简单到愚蠢)

Salt 状态系统的核心是 SLS，或者叫**S**a**L**t **S**tate 文件。SLS 表示系统将会是什么样的一种状态，而且是以一种很简单的格式来包含这些数据。这经常也被叫做配置管理。

所有都是数据

在深入研究细节之前，细节将有助于理解 SLS 文件只是蒙上面纱的数据结构。虽然明白 SLS 只是一种数据结构不是理解和使用 Salt States 的关键，但这对巩固知识是非常有用效的。

因此，SLS 文件实际上只是一些： 词典 *dictionaries*, 列表 *lists*, 字符串, and 数字.通过使用这些方法，使得 Salt 更加灵活. 多写 state 文件，将更加明白需要写什么。 其结果就是系统更加容易理解，即使系统管理员和开发人员的需求不断增加。

增加配置与用户