SaltStack简明教程

**SaltStack简介**

SaltStack是基于Python开发的一套C/S架构配置管理工具（功能不仅仅是配置管理，如使用salt-cloud配置AWS EC2实例），它的底层使用ZeroMQ消息队列pub/sub方式通信，使用SSL证书签发的方式进行认证管理。

号称世界上最快的消息队列ZeroMQ使得SaltStack能快速在成千上万台机器上进行各种操作，而且采用RSA Key方式确认身份，传输采用AES加密，这使得它的安全性得到了保障。

SaltStack经常被描述为Func加强版+Puppet精简版。



**为什么选择SaltStack？**

目前市场上主流的开源自动化配置管理工具有puppet、chef、ansible、saltstack等等。到底选择那个比较好？可以从以下几方面考虑：

**语言的选择（puppet/chef vs ansible/saltstack）**

Puppet、Chef基于Ruby开发，ansible、saltstack基于python开发的

运维开发语言热衷于python（后期可做二次开发），排除Puppet、Chef

**速度的选择 (ansible vs saltstack)**

ansible基于ssh协议传输数据，SaltStack使用消息队列zeroMQ传输数据。从网上数据来看，SaltStack比ansible快大约40倍。

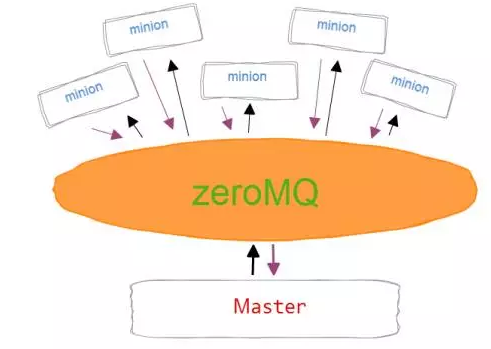
对比ansible，Saltstack缺点是需要安装客户端。为了速度建议选择SaltStack

SaltStack github地址：https://github.com/saltstack/salt

SaltStack官网文档地址：https://docs.saltstack.com

**SaltStack架构**

在SaltsStack架构中服务端叫作Master，客户端叫作Minion，都是以守护进程的模式运行，一直监听配置文件中定义的ret\_port（saltstack客户端与服务端通信的端口，负责接收客户端发送过来的结果，默认4506端口）和publish\_port（saltstack的消息发布系统，默认4505端口）的端口。当Minion运行时会自动连接到配置文件中定义的Master地址ret\_port端口进行连接认证。



1. Master：控制中心,salt命令运行和资源状态管理
2. Minion : 需要管理的客户端机器,会主动去连接Mater端,并从Master端得到资源状态
3. 信息,同步资源管理信息
4. States：配置管理的指令集
5. Modules：在命令行中和配置文件中使用的指令模块,可以在命令行中运行
6. Grains：minion端的变量,静态的
7. Pillar：minion端的变量,动态的比较私密的变量,可以通过配置文件实现同步minions定义
8. highstate：为minion端下发永久添加状态,从sls配置文件读取.即同步状态配置
9. salt\_schedule：会自动保持客户端配置

**SaltStack安装配置**

默认以CentOS6为例，采用yum安装，还有其它安装方式，如pip、源码、salt-bootstrap

**EPEL源配置**

1. rpm -ivh https://mirrors.tuna.tsinghua.edu.cn/epel/epel-release-latest-6.noarch.rpm

**安装、配置管理端(master)**

1. yum -y install salt-master
2. service salt-master start

注：需要iptables开启master端4505、4506端口

**安装被管理端(minion)**

1. yum -y install salt-minion
2. sed -i 's@#master:.\*@master: master\_ipaddress@' /etc/salt/minion  #master\_ipaddress为管理端IP
3. echo 10.252.137.141 > /etc/salt/minion\_id #个人习惯使用IP，默认主机名
4. service salt-minion start

**Master与Minion认证**

minion在第一次启动时，会在/etc/salt/pki/minion/（该路径在/etc/salt/minion里面设置）下自动生成minion.pem（private key）和 minion.pub（public key），然后将 minion.pub发送给master。master在接收到minion的public key后，通过salt-key命令accept minion public key，这样在master的/etc/salt/pki/master/minions下的将会存放以minion id命名的 public key，然后master就能对minion发送指令了。

**认证命令如下：**

1. [root@10.252.137.14 ~]# salt-key -L    #查看当前证书签证情况
2. Accepted Keys:
3. Unaccepted Keys:
4. 10.252.137.141
5. Rejected Keys:
6. [root@10.252.137.14 ~]# salt-key -A -y   #同意签证所有没有接受的签证情况
7. The following keys are going to be accepted:
8. Unaccepted Keys:
9. 10.252.137.141
10. Key **for** minion 10.252.137.141 accepted.
11. [root@10.252.137.14 ~]# salt-key -L
12. Accepted Keys:
13. 10.252.137.141
14. Unaccepted Keys:
15. Rejected Keys:

**SaltStack远程执行**

1. [root@10.252.137.14 ~]# salt '\*' test.ping
2. 10.252.137.141:
3. True
4. [root@10.252.137.14 ~]# salt '\*' cmd.run 'ls -al'
5. 10.252.137.141:
6. total 40
7. drwx------  4 root root 4096 Sep  7 15:01 .
8. drwxr-xr-x 22 root root 4096 Sep  3 22:10 ..
9. -rw-------  1 root root  501 Sep  7 14:49 .bash\_history
10. -rw-r--r--  1 root root 3106 Feb 20  2014 .bashrc
11. drwx------  2 root root 4096 Jan 30  2015 .cache
12. drwxr-xr-x  2 root root 4096 Apr 22 13:57 .pip
13. -rw-r--r--  1 root root  140 Feb 20  2014 .*profile*
14. -rw-r--r--  1 root root   64 Apr 22 13:57 .pydistutils.cfg
15. -rw-------  1 root root 4256 Sep  7 15:01 .viminfo

salt执行命令的格式如下：

1. salt '<target>' <function> [arguments]

target：执行salt命令的目标，可以使用正则表达式

function：方法，由module提供

arguments：function的参数

**target可以是以下内容：**

1. 正则表达式

1. salt -E 'Minion\*' test.ping  #主机名以Minion开通

2. 列表匹配

1. salt -L Minion,Minion1 test.ping

3. Grians匹配

1. salt -G '*os*:CentOS' test.ping

os:CentOS（默认存在）是Grains的键值对，数据以yaml保存在minion上，可在minion端直接编辑/etc/salt/grains，yaml格式。或者在master端执行salt '\*' grains.setval key "{'sub-key': 'val', 'sub-key2': 'val2'}" ,具体文档（命令salt \* sys.doc grains查看文档）

4. 组匹配

1. salt -N groups test.ping

如，在master新建/etc/salt/master.d/nodegroups.conf ，yaml格式

5. 复合匹配

1. salt -C 'G@*os*:CentOS **or** L@Minion' test.ping

6. Pillar值匹配

1. salt -I 'key:value' test.ping

/etc/salt/master设置pillar\_roots,数据以yaml保存在Master上

7. CIDR匹配

1. salt -S '10.252.137.0/24' test.ping

10.252.137.0/24是一个指定的CIDR网段

**function是module提供的方法**

通过下面命令可以查看所有的function：

1. salt '10.252.137.141' *sys*.doc *cmd*

function可以接受参数：

1. salt '10.252.137.141' *cmd*.run 'uname -a'

并且支持关键字参数：

1. #在所有minion上切换到/目录以salt用户运行uname -a命令。
2. salt '10.252.137.141' *cmd*.run 'uname -a' cwd=/ *user*=salt

**SaltStack配置管理**

**states文件**

salt states的核心是sls文件，该文件使用YAML语法定义了一些k/v的数据。

sls文件存放根路径在master配置文件中定义，默认为/srv/salt,该目录在操作系统上不存在，需要手动创建。

在salt中可以通过salt://代替根路径，例如你可以通过salt://top.sls访问/srv/salt/top.sls。

在states中top文件也由master配置文件定义，默认为top.sls，该文件为states的入口文件。

一个简单的sls文件如下：

1. apache:
2. pkg:
3. - installed
4. service:
5. - running
6. - require:
7. - pkg: apache

说明：此SLS数据确保叫做"apache"的软件包(package)已经安装,并且"apache"服务(service)正在运行中。

* 第一行，被称为ID说明(ID Declaration)。ID说明表明可以操控的名字。
* 第二行和第四行是State说明(State Declaration)，它们分别使用了pkg和service states。pkg state通过系统的包管理其管理关键包，service state管理系统服务(daemon)。 在pkg及service列下边是运行的方法。方法定义包和服务应该怎么做。此处是软件包应该被安装，服务应该处于运行中。
* 第六行使用require。本方法称为”必须指令”(Requisite Statement)，表明只有当apache软件包安装成功时，apache服务才启动起来。

state和方法可以通过点连起来，上面sls文件和下面文件意思相同。

1. apache:
2. pkg.installed
3. service.running
4. - require:
5. - pkg: apache

将上面sls保存为init.sls并放置在sal://apache目录下，结果如下：

1. /srv/salt
2. ├── apache
3. │   └── init.sls
4. └── top.sls

top.sls如何定义呢？

master配置文件中定义了三种环境，每种环境都可以定义多个目录，但是要避免冲突，分别如下：

1. # file\_roots:
2. #   base:
3. #     - /srv/salt/
4. #   dev:
5. #     - /srv/salt/dev/services
6. #     - /srv/salt/dev/states
7. #   prod:
8. #     - /srv/salt/prod/services
9. #     - /srv/salt/prod/states

top.sls可以这样定义：

1. base:
2. '\*':
3. - apache

说明：

第一行，声明使用base环境

第二行，定义target，这里是匹配所有

第三行，声明使用哪些states目录，salt会寻找每个目录下的init.sls文件。

**运行states**

一旦创建完states并修改完top.sls之后，你可以在master上执行下面命令：

1. [root@10.252.137.14 ~]# salt '\*' state.highstate
2. sk2:
3. ----------
4. State: - pkg
5. Name:      httpd
6. Function:  installed
7. Result:    True
8. Comment:   The following packages were installed/updated: httpd.
9. Changes:
10. ----------
11. httpd:
12. ----------
13. *new*:
14. 2.2.15-29.el6.centos
15. old:
16. ----------
17. State: - service
18. Name:      httpd
19. Function:  running
20. Result:    True
21. Comment:   Service httpd has been enabled, **and** **is** running
22. Changes:
23. ----------
24. httpd:
25. True
26. Summary
27. ------------
28. Succeeded: 2
29. Failed:    0
30. ------------
31. Total:     2

上面命令会触发所有minion从master下载top.sls文件以及其中定一个的states，然后编译、执行。执行完之后，minion会将执行结果的摘要信息汇报给master。

  原文来自：   https://blog.linuxeye.cn/450.html

附录(SaltStack编译安装)：

**自动化运维工具SaltStack详细部署**

2014-12-28 17:30:34

标签：[saltstack](http://blog.51cto.com/tag-saltstack.html)

原创作品，允许转载，转载时请务必以超链接形式标明文章 [原始出处](http://sofar.blog.51cto.com/353572/1596960) 、作者信息和本声明。否则将追究法律责任。<http://sofar.blog.51cto.com/353572/1596960>

==============================================================================

**基础介绍**

===============================================================================

**1、简介**

SaltStack是一个服务器基础架构集中化管理平台，具备配置管理、远程执行、监控等功能，一般可以理解为简化版的puppet和加强版的func。SaltStack基于Python语言实现，结合轻量级消息队列（ZeroMQ）与Python第三方模块（Pyzmq、PyCrypto、Pyjinjia2、python-msgpack和PyYAML等）构建。

通过部署SaltStack环境，我们可以在成千上万台服务器上做到批量执行命令，根据不同业务特性进行配置集中化管理、分发文件、采集服务器数据、操作系统基础及软件包管理等，SaltStack是运维人员提高工作效率、规范业务配置与操作的利器。

**2、特性**

(1)、部署简单、方便；

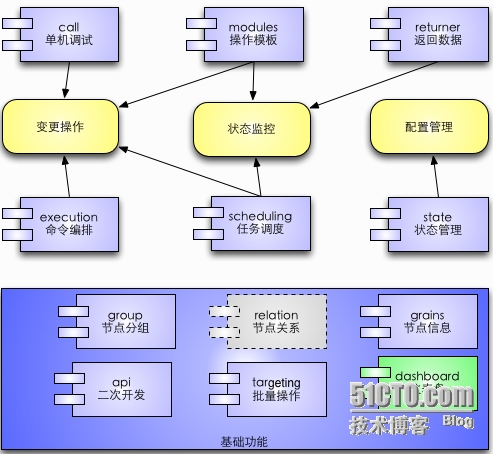
(2)、支持大部分UNIX/Linux及Windows环境；

(3)、主从集中化管理；

(4)、配置简单、功能强大、扩展性强；

(5)、主控端（master）和被控端（minion）基于证书认证，安全可靠；

(6)、支持API及自定义模块，可通过Python轻松扩展。

[](http://s3.51cto.com/wyfs02/M02/57/9A/wKiom1Sfzhmy2-OTAAIPxayI-Z4556.jpg)

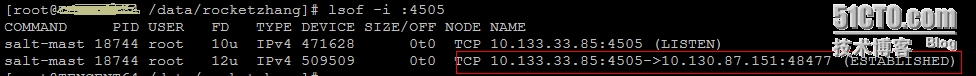
**3、Master与Minion认证**

(1)、minion在第一次启动时，会在/etc/salt/pki/minion/（该路径在/etc/salt/minion里面设置）下自动生成minion.pem（private key）和 minion.pub（public key），然后将 minion.pub发送给master。

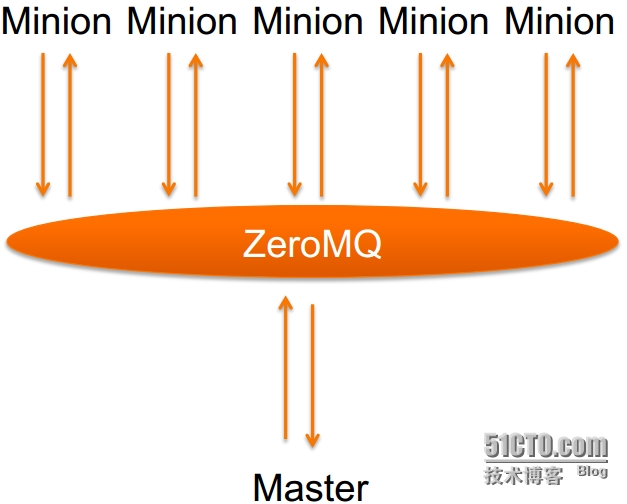
(2)、master在接收到minion的public key后，通过salt-key命令accept minion public key，这样在master的/etc/salt/pki/master/minions下的将会存放以minion id命名的 public key，然后master就能对minion发送指令了。

**4、Master与Minion的连接**

(1)、SaltStack master启动后默认监听4505和4506两个端口。4505（publish\_port）为saltstack的消息发布系统，4506（ret\_port）为saltstack客户端与服务端通信的端口。如果使用lsof 查看4505端口，会发现所有的minion在4505端口持续保持在ESTABLISHED状态。

[](http://s3.51cto.com/wyfs02/M00/57/97/wKioL1Sf0JHwhBhaAAEsYS9e6HE690.jpg)

(2)、minion与master之间的通信模式如下：

[](http://s3.51cto.com/wyfs02/M00/57/9A/wKiom1Sfzr-i8BUjAAF0EW4NxyY818.jpg)

**5、本次测试环境**

|  |  |  |
| --- | --- | --- |
| **IP地址** | **角色** | **系统环境** |
| 10.133.33.85 | master | CentOS 6.3 |
| 10.130.87.151 | minion | CentOS 6.3 |

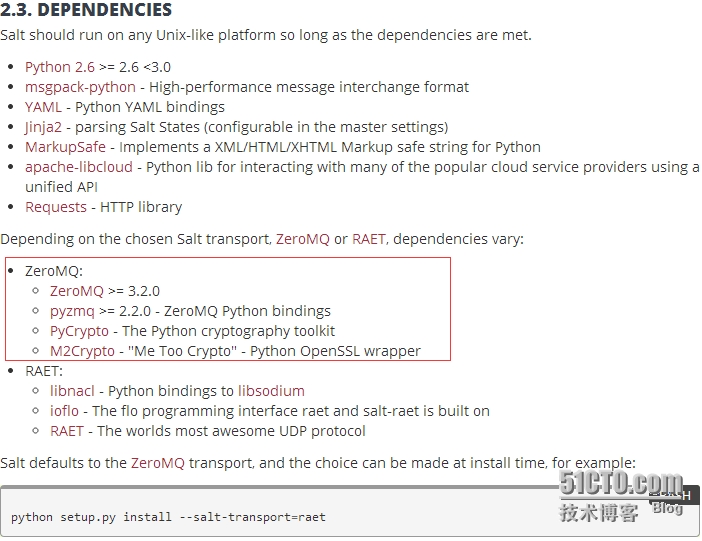
==========================================================================================

**二、SaltStack基础安装与配置**

==========================================================================================

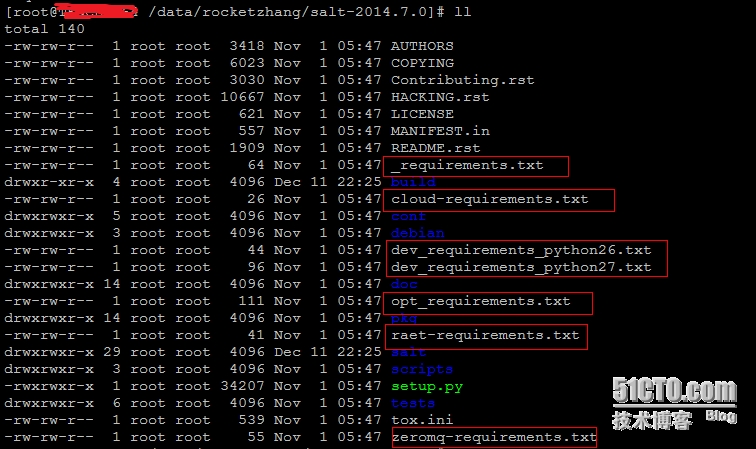
**1、依赖组件说明**

<http://docs.saltstack.com/en/latest/topics/installation/index.html>

[](http://s3.51cto.com/wyfs02/M02/57/97/wKioL1Sfz__ws2EVAANpFr96RiE332.jpg)

SaltStack的通信模式总共分为2种模式：ZeroMQ、REAT，鉴于REAT目前还不是太稳定，我们选择ZeroMQ模式，本次安装采用源码安装的方式。

解压源码后，这些文本文件里也有组件依赖相关说明哈：

[](http://s3.51cto.com/wyfs02/M01/57/97/wKioL1Sf0NaTTpjDAARykJLGBv0733.jpg)

**2、SaltStack基础安装**

**(1)、python 2.7安装**

<https://www.python.org/ftp/python/2.7.8/Python-2.7.8.tgz>

# tar xvzf Python-2.7.8.tgz

# cd Python-2.7.8

# ./configure --prefix=/usr/local

# make --jobs=`grep processor /proc/cpuinfo | wc -l`

# make install

## 将python头文件拷贝到标准目录，以避免编译saltstack时，找不到所需的头文件

# cd /usr/local/include/python2.7

# cp -a ./\* /usr/local/include/

## 备份旧版本的python，并符号链接新版本的python

# cd /usr/bin

# mv python python2.6

# ln -s /usr/local/bin/python

## 修改yum脚本，使其指向旧版本的python，已避免其无法运行

# vim /usr/bin/yum

#!/usr/bin/python  -->  #!/usr/bin/python2.6

**(2)、PyYAML模块安装**

<http://pyyaml.org/download/libyaml/yaml-0.1.5.tar.gz>

<https://pypi.python.org/packages/source/P/PyYAML/PyYAML-3.11.tar.gz>

# tar xvzf yaml-0.1.5.tar.gz

# cd yaml-0.1.5

# ./configure --prefix=/usr/local

# make --jobs=`grep processor /proc/cpuinfo | wc -l`

# make install

# tar xvzf PyYAML-3.11.tar.gz

# cd PyYAML-3.11

# python setup.py install

**(3)、setuptools模块安装**

<https://pypi.python.org/packages/source/s/setuptools/setuptools-7.0.tar.gz>

# tar xvzf setuptools-7.0.tar.gz

# cd setuptools-7.0

# python setup.py install

**(4)、markupsafe模块安装**

<https://pypi.python.org/packages/source/M/MarkupSafe/MarkupSafe-0.9.3.tar.gz>

# tar xvzf MarkupSafe-0.9.3.tar.gz

# cd MarkupSafe-0.9.3

# python setup.py install

**(5)、jinja2模块安装**

<https://pypi.python.org/packages/source/J/Jinja2/Jinja2-2.7.3.tar.gz>

# tar xvzf Jinja2-2.7.3.tar.gz

# cd Jinja2-2.7.3

# python setup.py install

**(6)、pyzmq模块安装**

<http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.gz>

注意：

系统自带的autoconf版本为2.63，在编译ZeroMQ时，由于版本较低，会报错不通过，故需安装大于2.63的版本。

# tar xvzf autoconf-2.69.tar.gz

# cd autoconf-2.69

# ./configure --prefix=/usr

# make

# make install

## 其他依赖库安装

# yum -y install libuuid.x86\_64 libuuid-devel.x86\_64

# yum -y install uuid.x86\_64 uuid-devel.x86\_64

# yum -y install uuid-c++.x86\_64 uuid-c++-devel.x86\_64

<https://github.com/jedisct1/libsodium/archive/1.0.1.tar.gz>

# tar xvzf libsodium-1.0.1.tar.gz

# cd libsodium-1.0.1

# ./autogen.sh

# ./configure --prefix=/usr/local

# make --jobs=`grep processor /proc/cpuinfo | wc -l`

# make install

<http://download.zeromq.org/zeromq-4.0.5.tar.gz>

<https://pypi.python.org/packages/source/p/pyzmq/pyzmq-14.4.1.tar.gz>

# tar xvzf zeromq-4.0.5.tar.gz

# cd zeromq-4.0.5

# ./autogen.sh

# ./configure --prefix=/usr/local

# make --jobs=`grep processor /proc/cpuinfo | wc -l`

# make install

# tar xvzf pyzmq-14.4.1.tar.gz

# cd pyzmq-14.4.1

# python setup.py configure --zmq=/usr/local

# python setup.py install

**(7)、M2Crypto模块安装**

# yum -y install swig.x86\_64

<http://www.openssl.org/source/openssl-1.0.1g.tar.gz>

# tar xvzf openssl-1.0.1g.tar.gz

# cd openssl-1.0.1g

# ./config shared --prefix=/usr/local

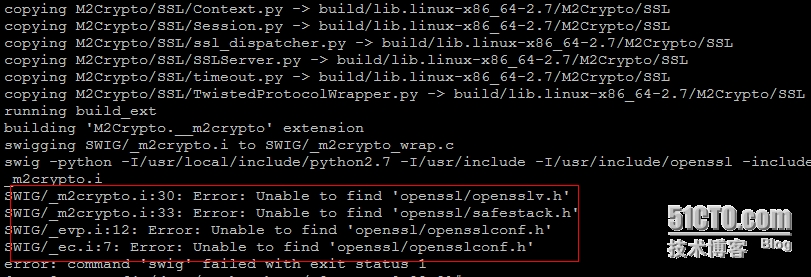
# make && make install

<https://pypi.python.org/packages/source/M/M2Crypto/M2Crypto-0.22.3.tar.gz>

# tar xvzf M2Crypto-0.22.3.tar.gz

# cd M2Crypto-0.22.3

# python setup.py install

注意：[](http://s3.51cto.com/wyfs02/M02/57/97/wKioL1Sf0SihxJdTAAOvt6zwKuQ140.jpg)

如果出现以上报错，执行以下命令：

# cp -a /usr/local/include/openssl /usr/include/

**(8)、msgpack模块安装**

<https://pypi.python.org/packages/source/m/msgpack-python/msgpack-python-0.4.2.tar.gz>

# tar xvzf msgpack-python-0.4.2.tar.gz

# cd msgpack-python-0.4.2

# python setup.py install

**(9)、msgpack-pure模块安装**

<https://pypi.python.org/packages/source/m/msgpack-pure/msgpack-pure-0.1.3.tar.gz>

# tar xvzf msgpack-pure-0.1.3.tar.gz

# cd msgpack-pure-0.1.3

# python setup.py install

**(10)、pycrypto模块安装**

<https://pypi.python.org/packages/source/p/pycrypto/pycrypto-2.6.1.tar.gz>

# tar xvzf pycrypto-2.6.1.tar.gz

# cd pycrypto-2.6.1

# python setup.py install

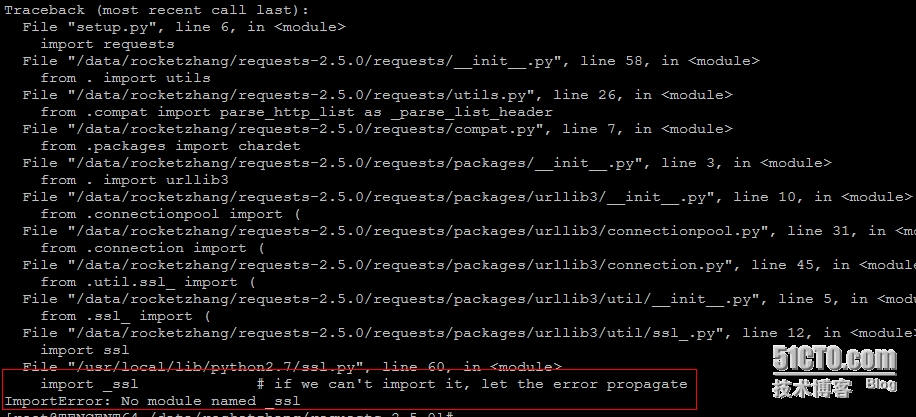
**(11)、requests模块安装**

<https://pypi.python.org/packages/source/r/requests/requests-2.5.0.tar.gz>

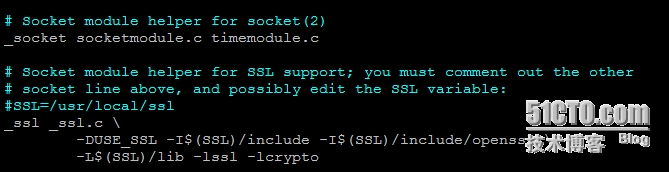
# tar xvzf requests-2.5.0.tar.gz

# cd requests-2.5.0

# python setup.py install

[](http://s3.51cto.com/wyfs02/M00/57/97/wKioL1Sf0ZnjB3EWAAT7h_5cIKk502.jpg)

# vim Python-2.7.8/Modules/Setup

[](http://s3.51cto.com/wyfs02/M01/57/9A/wKiom1Sf0SnAF2fRAAFMPwSeVfw093.jpg)

然后再重新对python-2.7.8进行编译安装即可解决以上问题。

# vim/etc/ld.so.conf

/usr/local/lib

/usr/local/lib64

# /sbin/ldconfig

# cd Python-2.7.8

# make && make install

注意：

如果openssl库在python之前已经安装，就不会出现这种情况，也无需经历此步骤，此处更多的是想说明一下组件之间的依赖关系，作为提醒。

**(12)、其他模块安装**

<https://pypi.python.org/packages/source/l/libnacl/libnacl-1.3.5.tar.gz>

<https://pypi.python.org/packages/source/i/ioflo/ioflo-1.0.2.tar.gz>

<https://pypi.python.org/packages/source/s/six/six-1.8.0.tar.gz>

<https://pypi.python.org/packages/source/r/raet/raet-0.4.2.tar.gz>

<https://pypi.python.org/packages/source/M/Mako/Mako-1.0.0.tar.gz>

注意：

以上模块为REAT通信模式所依赖的模块，作为可选项，非必要。

**(13)、saltstack安装**

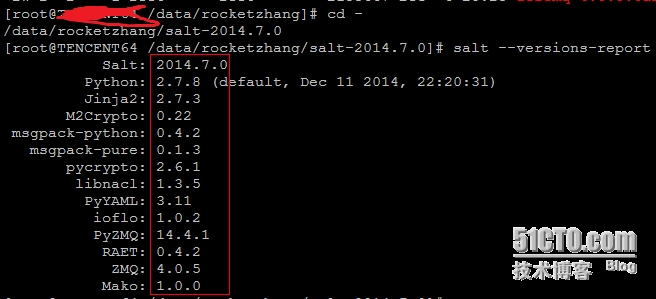
<https://github.com/saltstack/salt/releases/download/v2014.7.0/salt-2014.7.0.tar.gz>

# tar xvzf salt-2014.7.0.tar.gz

# cd salt-2014.7.0

# python setup.py install

# salt --versions-report

[](http://s3.51cto.com/wyfs02/M02/57/9A/wKiom1Sf0cKz9gDzAAGy2YGKprQ499.jpg)

**3、SaltStack配置**

**(1)、master端**

# mkdir /etc/salt

# cp -a conf/master /etc/salt/

# cp -a pkg/suse/salt-master /etc/init.d/

# chmod +x /etc/init.d/salt-master

# chkconfig --level 235 salt-master on

# mkdir -p /var/log/salt /srv/salt

# vim /etc/salt/master

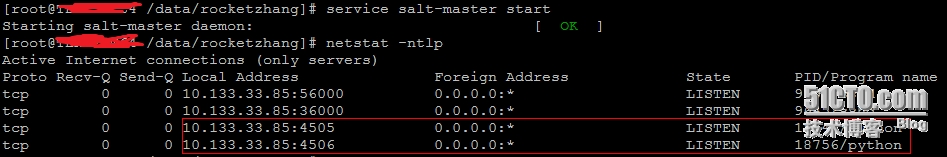
interface: 10.133.33.85

auto\_accept: True

# ln -s /usr/local/bin/salt-master /usr/bin/

# service salt-master start

# netstat -ntlp

[](http://s3.51cto.com/wyfs02/M01/57/9A/wKiom1Sf0kyT5tfmAAHOQhhowLw653.jpg)

# ps aux | grep python

[](http://s3.51cto.com/wyfs02/M00/57/9A/wKiom1Sf0sijET1tAAO0_Kr49_s600.jpg)

**(2)、minion端**

# mkdir /etc/salt

# cp -a conf/minion /etc/salt/

# cp -a pkg/suse/salt-minion /etc/init.d/

# chmod +x /etc/init.d/salt-minion

# chkconfig --level 235 salt-minion on

# mkdir -p /var/log/salt

# vim /etc/salt/minion

master: 10.133.33.85

id: minion-10-130-87-151

# ln -s /usr/local/bin/salt-minion /usr/bin/

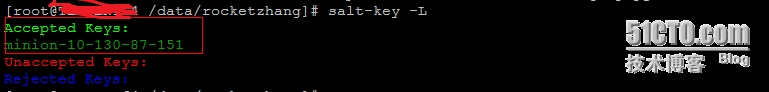
# service salt-minion start

# ps aux | grep python

[wKioL1Sf1AOhW4LqAADK39r9R4Y882.jpg](http://s3.51cto.com/wyfs02/M01/57/97/wKioL1Sf1AOhW4LqAADK39r9R4Y882.jpg)

**4、SaltStack简单使用**

## 查看当前的salt key信息

# salt-key -L[](http://s3.51cto.com/wyfs02/M01/57/9A/wKiom1Sf06XTHnz7AACYyu0j3cw532.jpg)

## 测试被控主机的连通性

# salt '\*' test.ping

[wKiom1Sf0-ujfcVrAAB3_5XnO4Q909.jpg](http://s3.51cto.com/wyfs02/M02/57/9A/wKiom1Sf0-ujfcVrAAB3_5XnO4Q909.jpg)

## 远程命令执行测试

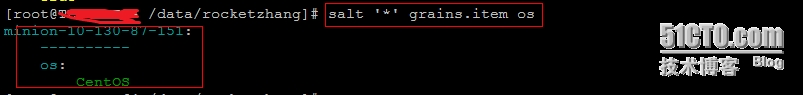
# salt '\*' cmd.run 'uptime'

[wKioL1Sf1PKhiFH3AAC4sHQ-dlk870.jpg](http://s3.51cto.com/wyfs02/M00/57/97/wKioL1Sf1PKhiFH3AAC4sHQ-dlk870.jpg)

## 根据被控主机的grains信息进行匹配过滤

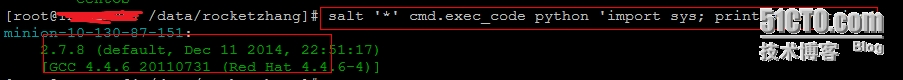
# salt -G 'os:Centos' test.ping[wKiom1Sf1JDyTqp4AAB63fXt6hw598.jpg](http://s3.51cto.com/wyfs02/M00/57/9A/wKiom1Sf1JDyTqp4AAB63fXt6hw598.jpg)

## 显示被控主机的操作系统类型

# salt '\*' grains.item os[](http://s3.51cto.com/wyfs02/M01/57/9A/wKiom1Sf1NTxlPsnAACcxbpeFK0008.jpg)

## 远程代码执行测试

# salt '\*' cmd.exec\_code python 'import sys; print sys.version'

[](http://s3.51cto.com/wyfs02/M02/57/97/wKioL1Sf1cWgTlOxAADseqP5EdA281.jpg)

**5、常用模块介绍**

(1)、cp模块（实现远程文件、目录的复制，以及下载URL文件等操作）

## 将主服务器file\_roots指定位置下的目录复制到被控主机

# salt '\*' cp.get\_dir salt://hellotest /data

##将主服务器file\_roots指定位置下的文件复制到被控主机

# salt '\*' cp.get\_file salt://hellotest/rocketzhang /root/rocketzhang

## 下载指定URL内容到被控主机指定位置

# salt '\*' cp.get\_url <http://xxx.xyz.com/download/0/files.tgz> /root/files.tgz

(2)、cmd模块（实现远程的命令行调用执行）

# salt '\*' cmd.run 'netstat -ntlp'

(3)、cron模块（实现被控主机的crontab操作）

## 为指定的被控主机、root用户添加crontab信息

# salt '\*' cron.set\_job root '\*/5' '\*' '\*' '\*' '\*' 'date >/dev/null 2>&1'

# salt '\*' cron.raw\_cron root

## 删除指定的被控主机、root用户的crontab信息

# salt '\*' cron.rm\_job root 'date >/dev/null 2>&1'

# salt '\*' cron.raw\_cron root

(4)、dnsutil模块（实现被控主机通用DNS操作）

## 为被控主机添加指定的hosts主机配置项

# salt '\*' dnsutil.hosts\_append /etc/hosts 127.0.0.1 rocketzhang.qq.com

(5)、file模块（被控主机文件常见操作，包括文件读写、权限、查找、校验等）

# salt '\*' file.get\_sum /etc/resolv.conf md5

# salt '\*' file.stats /etc/resolv.conf

更多功能可以看文档哈 ^\_^

(6)、network模块（返回被控主机网络信息）

# salt '\*' network.ip\_addrs

# salt '\*' network.interfaces

更多功能可以看文档哈 ^\_^

(7)、pkg包管理模块（被控主机程序包管理，如yum、apt-get等）

# salt '\*' pkg.install nmap

# salt '\*' pkg.file\_list nmap

(8)、service 服务模块（被控主机程序包服务管理）

# salt '\*' service.enable crond

# salt '\*' service.disable crond

# salt '\*' service.status crond

# salt '\*' service.stop crond

# salt '\*' service.start crond

# salt '\*' service.restart crond

# salt '\*' service.reload crond

(9)、更多功能

更多的功能，比如：grains、pillar、states、modules、returner、runners、reactor等，还有如下高级命令的使用，以及模板配置的渲染、扩展模块的二次开发等，可以自己去深入学习哈。

==========================================================================================

**附录**

==========================================================================================

一些学习资料：

<http://docs.saltstack.com/en/latest/ref/modules/all/index.html>

<http://docs.saltstack.com/en/latest/ref/modules/index.html>

<http://www.saltstack.cn/projects/cssug-kb/wiki>

<http://salt-api.readthedocs.org/en/latest/>

<http://xiaorui.cc/category/saltstack/>

[http://www.shencan.net/index.php/category/%e8%87%aa%e5%8a%a8%e5%8c%96%e8%bf%90%e7%bb%b4/saltstack/](http://www.shencan.net/index.php/category/%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%90%E7%BB%B4/saltstack/)

<http://liuping0906.blog.51cto.com/2516248/d-13>

<http://lixcto.blog.51cto.com/4834175/d-1>

<http://blog.coocla.org/category/saltstack>

<http://pengyao.org/>

资源站点：

<https://github.com/saltstack/>

<https://github.com/saltstack-formulas>

<https://github.com/cssug>