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，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。

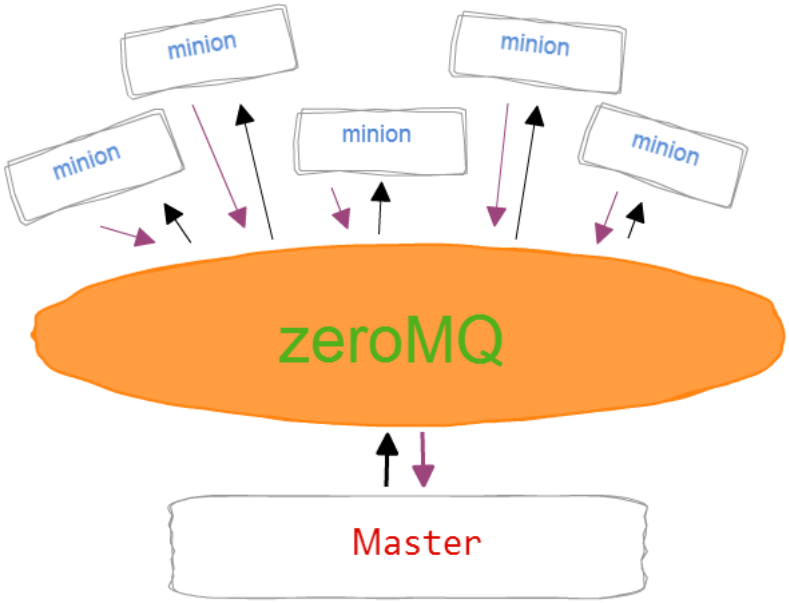
对比ansible，Saltstack缺点是需要安装客户端。为了速度建议选择SaltStack

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.
```

```
[root@10.252.137.14 ~]# salt-key -L
Accepted Keys:
10.252.137.141
Unaccepted Keys:
Rejected Keys:
```

SaltStack远程执行

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

salt执行命令的格式如下：

```
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

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

2. 列表匹配

```
salt -L Minion,Minion1 test.ping
```

3. Grians匹配

```
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

```
salt -N groups test.ping
```

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

5. 复合匹配

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

6. Pillar值匹配

```
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

```
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：

```
salt '10.252.137.141' sys.doc cmd
```

function可以接受参数：

```
salt '10.252.137.141' cmd.run 'uname -a'
```

并且支持关键字参数：

```
#在所有minion上切换到/目录以salt用户运行uname -a命令。
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：

```
apache:
 pkg:
   - installed
 service:
   - running
   - require:
     - 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。

```
apache:
 pkg.installed
 service.running
   - require:
     - pkg: apache
```

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

```
/srv/salt
├── apache
│   └── init.sls
└── 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：

```
# file_roots:
#   base:
#     - /srv/salt/
#   dev:
#     - /srv/salt/dev/services
#     - /srv/salt/dev/states
#   prod:
#     - /srv/salt/prod/services
#     - /srv/salt/prod/states
```

top.sls可以这样定义：

```
base:
  '*':
    - 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：

```
[root@10.252.137.14 ~]# salt '*' state.highstate
sk2:
----------
State: - pkg
Name:      httpd
Function:  installed
Result:    True
Comment:   The following packages were installed/updated: httpd.
Changes:
----------
httpd:
----------
new:
2.2.15-29.el6.centos
old:

----------
State: - service
Name:      httpd
Function:  running
Result:    True
Comment:   Service httpd has been enabled, and is running
Changes:
----------
httpd:
True

Summary
------------
Succeeded: 2
Failed:    0
------------
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。

Wed Sep 7 16:45:45 CST 20