salt-api安装配置及使用

Python3使用saltstack和salt-api

安装python3

```
1. tar zxvf Python-3.5.1.tgz
2. cd  Python-3.5.1
3. ./configure
4. make
5. make install
6. mv  /usr/bin/python /usr/bin/python2 # 如果是软连接，可以直接删除
7. ln -s /usr/local/bin/python3.5 /usr/bin/python
8. vim /usr/bin/yum   # 修改Yum,使yum依然有效，yum依靠老版本的python
9. #!/usr/bin/python 修改为#!/usr/bin/python2

# 修改完/usr/bin/yum 依然还有问题，可以尝试修改/usr/libexec/urlgrabber-ext-down的文件python抬头
```

安装 salt-api

```
yum install salt-api -y
```

配置

- 生成自签名证书(用于ssl)

```
cd  /etc/pki/tls/certs
# 生成自签名证书,过程中需要输入key密码及RDNs
make testcert
cd /etc/pki/tls/private/
# 解密key文件，生成无密码的key文件,过程中需要输入key密码，该密码为之前生成证书时设置的密码
openssl rsa -in localhost.key -out localhost_nopass.key
```

- 创建用于salt-api的用户

```
useradd -M -s /sbin/nologin salt-api
echo "salt-api" | passwd salt-api —stdin
```

- 修改/etc/salt/master文件

```
sed -i '/#default_include/s/#default/default/g'
/etc/salt/master
mkdir /etc/salt/master.d
```

- 新增配置文件/etc/salt/master.d/api.conf

```
cat /etc/salt/master.d/api.conf
rest_cherrypy:
  port: 8000
  ssl_crt: /etc/pki/tls/certs/localhost.crt
  ssl_key: /etc/pki/tls/private/localhost_nopass.key
```

- 新增配置文件/etc/salt/master.d/eauth.conf

```
cat /etc/salt/master.d/eauth.conf
external_auth:
  pam:
    salt-api:
      - .*
      - '@wheel'
      - '@runner'
```

- 启动salt-master and salt-api

```
systemctl start salt-master
systemctl start salt-api
```

- 安装一个salt client

```
yum install salt-minion -y
修改配置
```

```
sed -i "/^#master: salt/c master: 192.168.104.76"
/etc/salt/minion
启动 client
systemctl start salt-minion
```

- master 上接受key

```
[root@node76 salt]# salt-key -L
Accepted Keys:
Denied Keys:
Unaccepted Keys:
node76
Rejected Keys:
[root@node76 salt]# salt-key -A
The following keys are going to be accepted:
Unaccepted Keys:
node76
Proceed? [n/Y] Y
Key for minion node76 accepted.
[root@node76 salt]# salt-key -L
Accepted Keys:
node76
Denied Keys:
Unaccepted Keys:
Rejected Keys:
```

api使用

- 使用curl 获取token

```
 curl -k https://192.168.104.76:8000/login -H "Accept:
application/x-yaml"  -d username='salt-api' -d password='salt-api'
-d eauth='pam'
return:
- eauth: pam
```

```
expire: 1520269544.2591
perms:
- .*
- '@wheel'
- '@runner'
start: 1520226344.259099
token: 593a7224f988f28b84d58b7cda38fe5e5ea07d98
user: salt-api
```

获取token后就可以使用token通信

==注==：重启salt-api后token改变

- 测试minion端的联通性

下面功能类似于“salt '*' test.ping”

```
curl -k https://192.168.104.76:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token: ded897184a942ca75683276c29d787ea71c207a9" -d client='local' -d tgt='*' -d fun='test.ping'
return:
- node76: true
```

- 参数解释:

client : 模块，python处理salt-api的主要模块，‘client interfaces <netapi-clients>’
local : 使用‘LocalClient <salt.client.LocalClient>’ 发送命令给受控主机，等价于saltstack命令行中的'salt'命令
local_async : 和**local**不同之处在于，这个模块是用于异步操作的，即在master端执行命令后返回的是一个jobid，任务放在后台运行，通过产看jobid的结果来获取命令的执行结果。
runner : 使用'RunnerClient<salt.runner.RunnerClient>' 调用salt-master上的runner模块，等价于saltstack命令行中的'salt-run'命令

```
runner_async : 异步执行runner模块
wheel : 使用'WheelClient<salt.wheel.WheelClient>', 调用salt-master上的wheel模块，wheel模块没有在命令行端等价的模块，但它通常管理主机资源，比如文件状态，pillar文件，salt配置文件，以及关键模块<salt.wheel.key>功能类似于命令行中的salt-key。
wheel_async : 异步执行wheel模块
备注：一般情况下local模块，需要tgt和arg(数组)，kwarg(字典)，因为这些值将被发送到minions并用于执行所请求的函数。而runner和wheel都是直接应用于master，不需要这些参数。
tgt : minions
fun : 函数
arg : 参数
expr_form : tgt的匹配规则
'glob' - Bash glob completion - Default
'pcre' - Perl style regular expression
'list' - Python list of hosts
'grain' - Match based on a grain comparison
'grain_pcre' - Grain comparison with a regex
'pillar' - Pillar data comparison
'nodegroup' - Match on nodegroup
'range' - Use a Range server for matching
'compound' - Pass a compound match string
```

- 执行远程命令

下面功能类似于“salt '*' cmd.run ifconfig”

```
curl -k https://192.168.104.76:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token: ded897184a942ca75683276c29d787ea71c207a9" -d client='local' -d tgt='*' -d fun='cmd.run'   -d arg='uptime'
return:
- node76: ' 13:18:46 up 161 days,  2:23,  1 user,  load average: 0.15, 0.09, 0.10'
```

- 使用state.sls

下面功能类似于“salt '*' state.sls ifconfig”

```
 curl -k https://192.168.104.76:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token: ded897184a942ca75683276c29d787ea71c207a9" -d client='local' -d tgt='*' -d fun='state.sls' -d arg='ifconfig'
return:
- node76:
        cmd_|-ifconfig_|-ifconfig_|-run:
      __run_num__: 0
      changes:
        pid: 30954
        retcode: 0
        stderr: ''
        stdout: "eth2      Link encap:Ethernet  HWaddr 00:50:56:B5:5C:28  \n     \
      \     inet addr:192.168.90.63  Bcast:192.168.90.255  Mask:255.255.255.0\n\
      \          inet6 addr: fe80::250:56ff:feb5:5c28/64 Scope:Link\n\
      \  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1\n          RX packets:825051\
      \ errors:0 dropped:0 overruns:0 frame:0\n          TX packets:434351 errors:0\
      \ dropped:0 overruns:0 carrier:0\n          collisions:0 txqueuelen:1000\
      \ \n          RX bytes:60353823 (57.5 MiB)  TX bytes:27062672 (25.8 MiB)\n\
      \nlo        Link encap:Local Loopback  \n          inet addr:127.0.0.1 \
      \ Mask:255.0.0.0\n          inet6 addr: ::1/128 Scope:Host\n          UP\
      \ LOOPBACK RUNNING  MTU:16436  Metric:1\n          RX
packets:808 errors:0\
      \ dropped:0 overruns:0 frame:0\n          TX packets:808
errors:0 dropped:0\
      \ overruns:0 carrier:0\n          collisions:0 txqueuelen:0 \n
\
      \ RX bytes:59931 (58.5 KiB)  TX bytes:59931 (58.5 KiB)"
      comment: Command "ifconfig" run
      duration: 11.991
      name: ifconfig
      result: true
      start_time: '13:59:06.334112'
```

- 使用Targeting

下面功能类似于"salt -L '192.168.90.61,192.168.90.63' test.ping"

```
 curl -k https://192.168.104.76:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token:
ded897184a942ca75683276c29d787ea71c207a9"  -d
client='local' -d tgt='node76'  -d expr_form='list'  -d
fun='test.ping'
return:
- node76: true
```

- 以json格式输出

```
curl -k https://192.168.104.76:8000 -H "Accept:
application/json" -H "X-Auth-Token:
ded897184a942ca75683276c29d787ea71c207a9"  -d
client='local' -d tgt='node76'  -d fun='cmd.run' -d
arg='uptime'
{"return": [{"node76": " 13:25:20 up 161 days,  2:30,  1 user,
load average: 0.01, 0.06, 0.08"}]}
```