saltstack grains组件笔记整理

环境需求：

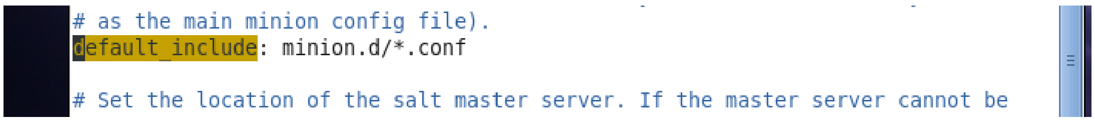
readhat 6.5

saltstack master和minion端各一个（可以在一台电脑上）

salt grains 组件配置

1、在minion端进行配置

1、配置minion配置文件中的default_include:为minion.d. 位置：/etc/salt/minion



```
# as the main minion config file).
default_include: minion.d/*.conf

# Set the location of the salt master server. If the master server cannot be
```

2、在/etc/salt/minion.d下定义一个hostinfo.conf

并且输入一些配置：

grains:

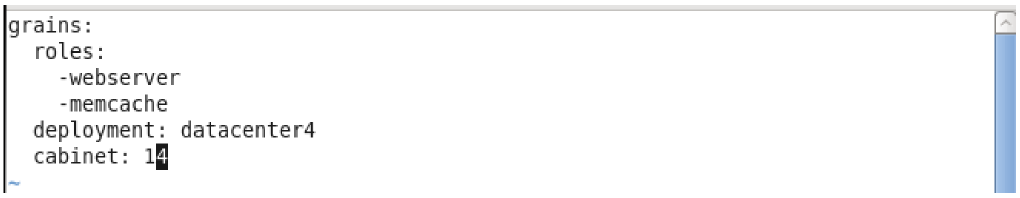
roles:

-webserver

-memcache

deployment: datacenter4

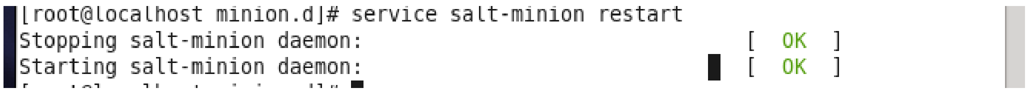
cabinet: 13



```
grains:
  roles:
    -webserver
    -memcache
  deployment: datacenter4
  cabinet: 14
~
```

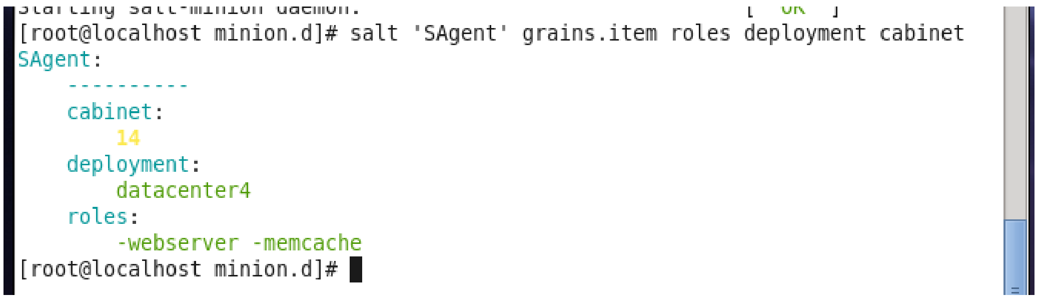
然后重新启动客户端服务

service salt-minion restart



```
[root@localhost minion.d]# service salt-minion restart
Stopping salt-minion daemon:                               [  OK  ]
Starting salt-minion daemon:                               [  OK  ]
```

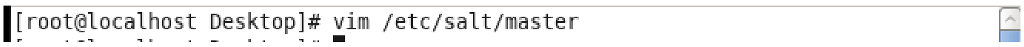
我们使用salt 'SAgent' grains.item roles deployment cabinet来看看我们配置的效果



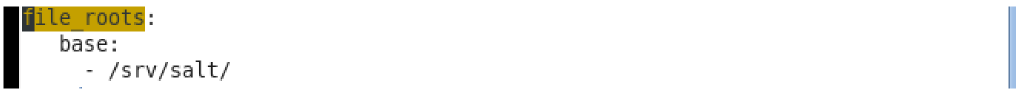
```
[root@localhost minion.d]# salt 'SAgent' grains.item roles deployment cabinet
SAgent:
    ----------
    cabinet:
        14
    deployment:
        datacenter4
    roles:
        -webserver -memcache
[root@localhost minion.d]# 
```

2、在master端进行脚本配置

1、查看一下我们的base目录（就是在master主控制文件file_roots指出的路径）



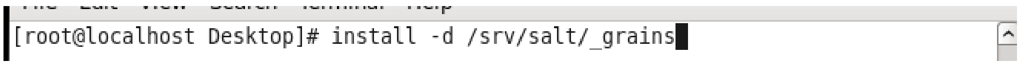
```
[root@localhost Desktop]# vim /etc/salt/master
```



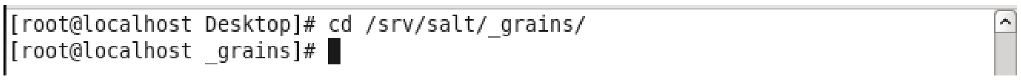
```
file_roots:
  base:
    - /srv/salt/
```

2、创建我们要写python脚本的目录

install -d /srv/salt/_grains

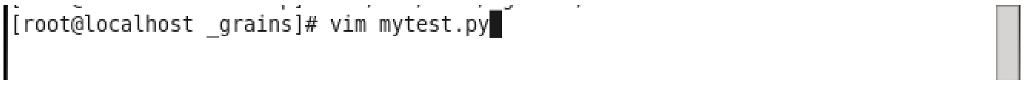


```
[root@localhost Desktop]# install -d /srv/salt/_grains
```



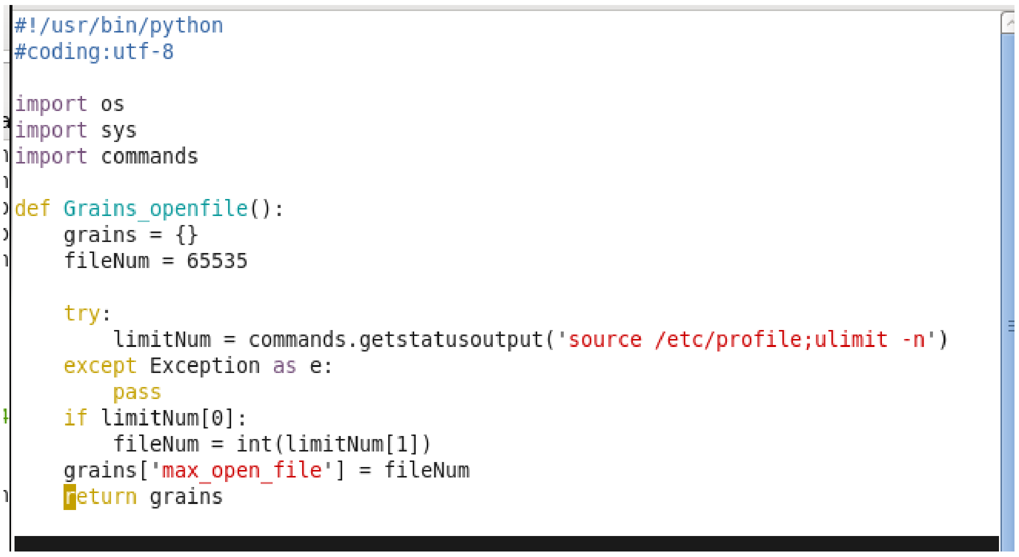
```
[root@localhost Desktop]# cd /srv/salt/_grains/
[root@localhost _grains]#
```

3、脚本开始



```
[root@localhost _grains]# vim mytest.py
```

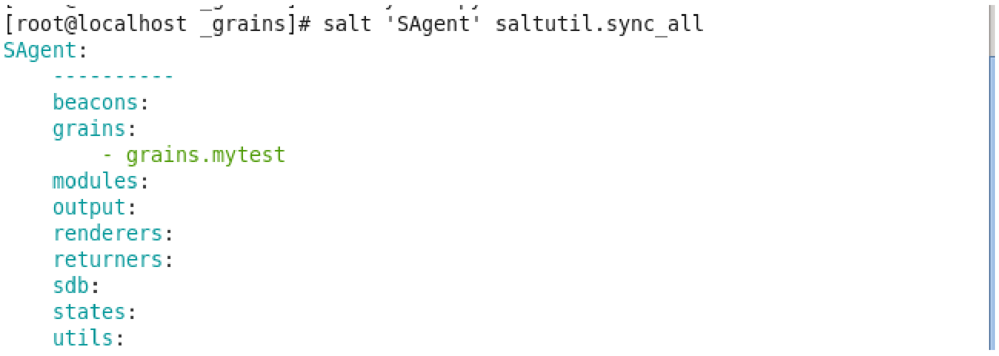
```
#!/usr/bin/python
#coding:utf-8

import os
import sys
import commands

def Grains_openfile():
    grains = {}
    fileNum = 65535

    try:
        limitNum = commands.getstatusoutput('source /etc/profile;ulimit -n')
    except Exception as e:
        pass
    if limitNum[0]:
        fileNum = int(limitNum[1])
    grains['max_open_file'] = fileNum
    return grains
```



```
#!/usr/bin/python
#coding:utf-8

import os
import sys
import commands

def Grains_openfile():
    grains = {}
    fileNum = 65535

    try:
        limitNum = commands.getstatusoutput('source /etc/profile;ulimit -n')
    except Exception as e:
        pass
    if limitNum[0]:
        fileNum = int(limitNum[1])
    grains['max_open_file'] = fileNum
    return grains
```

3、同步脚本到制定的minion

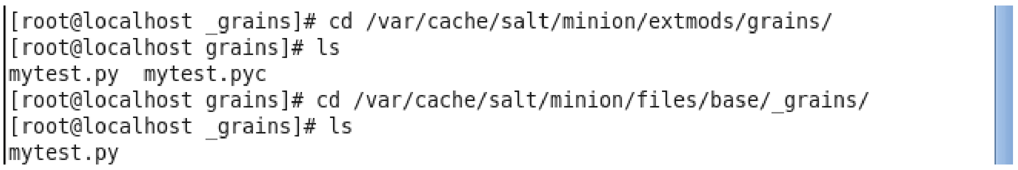
salt 'SAgent' saltutil.sync_all



```
[root@localhost _grains]# salt 'SAgent' saltutil.sync_all
SAgent:
    ----------
    beacons:
    grains:
        - grains.mytest
    modules:
    output:
    renderers:
    returners:
    sdb:
    states:
    utils:
```

4、查看效果

1、首先我们minion的目录下有我们写的脚本了

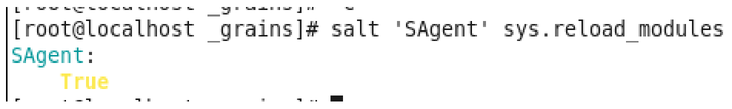


```
[root@localhost _grains]# cd /var/cache/salt/minion/extmods/grains/
[root@localhost grains]# ls
mytest.py  mytest.pyc
[root@localhost grains]# cd /var/cache/salt/minion/files/base/_grains/
[root@localhost _grains]# ls
mytest.py
```

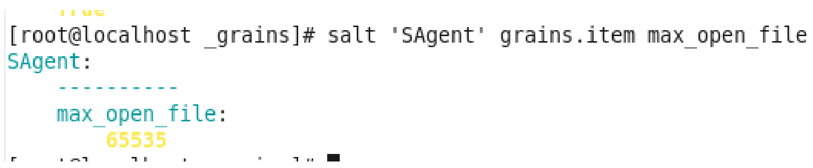
/var/cache/salt/minion/extmods/grains/　扩展文件的最终存放位置

/var/cache/salt/minion/files/base/_grains/　扩展文件的临时存放位置

1、刷新模块然后看看效果



```
[root@localhost _grains]# salt 'SAgent' sys.reload_modules
SAgent:
    True
```



```
[root@localhost _grains]# salt 'SAgent' grains.item max_open_file
SAgent:
    ----------
    max_open_file:
        65535
```