```
salt-api

Salt-api是Master和Minion之外的一个独立服务，所以需要独立部署，API服务需要部署在Master服务器上.

1. 安装salt-api
   yum -y install salt-api
   service salt-api restart
   chkconfig  salt-api on

2.启用salt-master配置文件夹
   取消 default_include: master.d/*.conf  注释
   mkdir -p /etc/salt/master.d

3.创建调用Api时候的账号
  1) useradd -M -s /sbin/nologin saltapi
  2) echo 'saltapi' | passwd saltapi --stdin

4.创建saltApi配置文件

[root@k8s-master1 master.d]# systemctl restart salt-api

[root@localhost master.d]# cat api.conf
rest_cherrypy:
  port: 8000  ##监听端口
  host: 0.0.0.0  ##在所有地址上
  disable_ssl: True  ##关闭SSL
  debug: True

[root@localhost master.d]# cat eauth.conf
external_auth:
  pam:   ##认证模式， PAM代表用Linux本身的用户进行身份认证  salt还支持其他的认证类型
    saltapi:    ##使用saltapi账号
      - .*   ###此处代表所有权限
       - '@wheel'   ##salt-api 支持wheel
       - '@runner'  ##salt-api 支持salt-run


5.启动服务 查看端口监听状态
[root@localhost master.d]# ps -ef | grep salt-api
root      30380      1  0 01:07 ?        00:00:00 /usr/bin/python2.7 /usr/bin/salt-api -d
root      30383  30380  1 01:07 ?        00:00:10 /usr/bin/python2.7 /usr/bin/salt-api -d
root      30911   8467  0 01:18 pts/2    00:00:00 grep salt-api

[root@localhost master.d]# netstat -tuan | grep 8000
tcp        0      0 0.0.0.0:8000                0.0.0.0:*                   LISTEN


6.申请token
[root@localhost master.d]# curl -k http://172.16.70.231:9000/login -H "Accept: application/x-yaml"  -d username='saltapi' -d password='saltapi'  -d eauth='pam'

return:
- eauth: pam
  expire: 1524287969.035184
  perms:
  - .*
  start: 1524244769.035183
  token: 4e19395bc1721a563a2f711acc1fd60c05b5d257 ###后续的请求都需要用这个token
  user: saltapi


7.通过接口请求

1)salt  '*' test.ping

[root@localhost master.d]# curl -k http://192.168.111.130:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token: 033ec1f6703e9d41cc830d48ff6cfb3c50ce5c9e" -d
client='local' -d tgt='*' -d fun='test.ping'
return:
- 192.168.111.131: true
  192.168.111.132: true

2) salt  '*'  cmd.run ifconfig
[root@localhost master.d]# curl -k http://192.168.111.130:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token: 033ec1f6703e9d41cc830d48ff6cfb3c50ce5c9e" -d
client='local' -d tgt='*' -d fun='cmd.run' -d arg='ifconfig'

注：client='local' 代表的是 salt 命令


client参数详解：  client模块，Python处理salt-api的主要模块，‘client interfaces <netapi-clients>’
 1     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
 2     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。
 3     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
 4     runner_async ：异步执行runner模块
 5     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。
 6     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



return:
- 192.168.111.131: "eth0      Link encap:Ethernet  HWaddr 00:0C:29:5B:AE:2E  \n          inet
    addr:192.168.111.131  Bcast:192.168.111.255  Mask:255.255.255.0\n          inet6
    addr: fe80::20c:29ff:fe5b:ae2e/64 Scope:Link\n          UP BROADCAST RUNNING MULTICAST
    \ MTU:1500  Metric:1\n          RX packets:85460 errors:0 dropped:0 overruns:0
    frame:0\n          TX packets:36826 errors:0 dropped:0 overruns:0 carrier:0\n
    \         collisions:0 txqueuelen:1000 \n          RX bytes:72827828 (69.4 MiB)
    \ TX bytes:4990587 (4.7 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:184 errors:0 dropped:0
    overruns:0 frame:0\n          TX packets:184 errors:0 dropped:0 overruns:0 carrier:0\n
    \         collisions:0 txqueuelen:0 \n          RX bytes:11452 (11.1 KiB)  TX
    bytes:11452 (11.1 KiB)"
  192.168.111.132: "eth0      Link encap:Ethernet  HWaddr 00:0C:29:22:07:48  \n          inet
    addr:192.168.111.132  Bcast:192.168.111.255  Mask:255.255.255.0\n          inet6
    addr: fe80::20c:29ff:fe22:748/64 Scope:Link\n          UP BROADCAST RUNNING MULTICAST
    \ MTU:1500  Metric:1\n          RX packets:80978 errors:0 dropped:0 overruns:0
    frame:0\n          TX packets:33130 errors:0 dropped:0 overruns:0 carrier:0\n
    \         collisions:0 txqueuelen:1000 \n          RX bytes:72097198 (68.7 MiB)
    \ TX bytes:4070712 (3.8 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:174 errors:0 dropped:0
```

```
    overruns:0 frame:0\n          TX packets:174 errors:0 dropped:0 overruns:0 carrier:0\n
    \         collisions:0 txqueuelen:0 \n          RX bytes:9372 (9.1 KiB)  TX bytes:9372
    (9.1 KiB)"

##综上可见  saltstack  restapi  即可成功使用

注：将结果写入elasticsearch
curl -k http://172.16.70.231:9000 -H "Accept: application/json" -H "X-Auth-Token: b39ecc28b2f59bab08ff8326be22ab7a7956b83e" -d client='local' -d tgt='172.16.70.233'
-d fun='sys_init.memory_load' -d ret='elasticsearch'

调用saltapi时的参数类型
#https://docs.saltstack.com/en/latest/ref/clients/ 官方文档路径

cmd(tgt, fun, arg=(), timeout=None, tgt_type=u'glob', ret=u'', jid=u'', full_return=False, kwarg=None, **kwargs)

参数类型：
1) tgt (string or list) -- 要执行命令的目标主机，默认类型是字符串，如果要在多台主机上执行命令，tgt的值为列表，同时要修改tgt type选项

2) tgt_type --
    The type of tgt. Allowed values:

    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
    pillar_pcre - Pillar data comparison with a regex
    nodegroup - Match on nodegroup
    range - Use a Range server for matching
    compound - Pass a compound match string
    ipcidr - Match based on Subnet (CIDR notation) or IPv4 address.
3) timeout -- 设置命令超时时间

4) arg (list or list-of-lists) -- 远程主机上执行的函数的参数
```

In [3]:
```
#示例代码:Python调用salt-api

#!/usr/bin/python
#coding:utf8


import os
import requests
import time
from util import *

proj_dir = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
config_dir = os.path.join(proj_dir, 'config')
config_file = os.path.join(config_dir, 'config.cfg')

saltConfig = readConfig(config_file).read_config("saltstack")


class RequestError(Exception):
    pass

class saltApi:
    tokenFile = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'saltApiToken.file')
    username = saltConfig["username"]
    password = saltConfig["password"]
    eauth = saltConfig["eauth"]
    def __init__(self, saltMaster=None, saltMasterPort=9000, saltMinion=None, saltFunc = None, saltCmd = None):
        self.saltMaster = saltMaster
        self.saltMinionList = saltMinion.split(',')
        self.saltCmd = saltCmd
        self.saltFunc = saltFunc
        self.saltMasterPort=saltMasterPort

    def __dateToday(self, _format='%Y%m%d %H:%M:%S'):
        a = time.strftime(_format, time.localtime())
        return int(time.mktime(time.strptime(a, _format)))

    def __saveToken(self, kwargs):
        ##kwargs  {"master": {"token", "deadTime"}}
        existToken = self.__readToken()
        for (k, v) in kwargs.items():
            existToken[k] = v
        f = open(saltApi.tokenFile, 'w+')
        for (k, v) in existToken.items():
            line = ' '.join([k, v.get('token', ''), v.get('deadTime', '')]) + '\r\n'
            f.write(line)
            f.flush()
        f.close()

    def __readToken(self):
        result = {}
        if not os.path.exists(saltApi.tokenFile):
            f = open(saltApi.tokenFile, 'w+')
            f.close()  ##创建token文件
        with open(saltApi.tokenFile) as file:
            for line in file:
                sp = line.strip().split()
                if len(sp) != 3:
                    continue
                result.setdefault(sp[0], {})['token'] = sp[1]
                result.setdefault(sp[0], {})['deadTime'] = sp[2]
        return result   #{"192.168.1.1":{"token": 'xx', "deadTime": 11111}}

    def getToken(self):
        result = {}
        headers = {'Accept': 'application/json'}
        url = '%s://%s:%s/login' %(saltConfig["web"], self.saltMaster,self.saltMasterPort)
        data = {'username':saltApi.username, 'password': saltApi.password, 'eauth': saltApi.eauth}
        try:
            response = requests.post(url=url, headers=headers, json=data)
        except Exception as e:
            result[self.saltMaster] = e
            result.setdefault(self.saltMaster, {})['token'] = "token Error!"
            result.setdefault(self.saltMaster, {})['deadTime'] = " "
        else:
            if response.status_code == 200:
                token = str(response.json()['return'][0].get('token', ''))
                deadTime = str(response.json()['return'][0].get('expire', ''))
                result.setdefault(self.saltMaster, {})['token'] = token
                result.setdefault(self.saltMaster, {})['deadTime'] = deadTime
            else:
                raise RequestError("saltApi Token get Error!! %s" %(response.status_code))
            self.__saveToken(result)
        return result

    def saltCmdPing(self):
        result = {}
        masToken = self.__readToken().get(self.saltMaster, {}).get('token')
        headers = {'Accept': 'application/json', "X-Auth-Token": masToken}
        url = "%s://%s:%s" %(saltConfig["web"], self.saltMaster, self.saltMasterPort)
        for min in self.saltMinionList:
            data = {"tgt": min, "fun": "test.ping", "client": 'local'}
            try:
                response = requests.post(url=url, headers=headers, json=data)
            except Exception as e:
                tmp = {}
                tmp.setdefault(min, {})["status"] = False
                tmp.setdefault(min, {})["comment"] = "%s saltApi Request Failed!! %s" %(min, e)
                result.setdefault(self.saltMaster, {}).update(tmp)
            else:

                if response.status_code == 200:
                    pingStatus = response.json()["return"][0].get(min, False)
                    tmp = {}
                    tmp.setdefault(min, {})["status"] = pingStatus
                    tmp.setdefault(min, {})['comment'] = '%s SaltPing Successfully!' %min
                    if not pingStatus:
                        tmp.setdefault(min, {})["comment"] = "%s SaltPing Failed!!" %min
                    result.setdefault(self.saltMaster, {}).update(tmp)
                elif response.status_code == 401:
                    self.getToken()
                    return self.saltCmdPing()
                else:
                    tmp = {}
                    tmp.setdefault(min, {})["status"] = False
                    tmp.setdefault(min, {})["comment"] = "%s saltApi Http Request unknown Failed!! %s" %(min, response.text)
                    result.setdefault(self.saltMaster, {}).update(tmp)
        return result

    def saltCmdExecute(self, minion, token):
        result = {}
        url = "%s://%s:%s" % (saltConfig["web"], self.saltMaster, self.saltMasterPort)
        headers = {'Accept': 'application/json', "X-Auth-Token": token}
        data = {"tgt": minion, "fun": self.saltFunc, "client": 'local', "arg": self.saltCmd}
        try:
            response = requests.post(url=url, headers=headers, json=data)
```

```
        except:
            result.setdefault(self.saltMaster, {}).setdefault(minion, {})["status"] = False
            result.setdefault(self.saltMaster, {}).setdefault(minion, {})["comment"] = "Salt Cmd Http Request error!!!"
        else:
            if response.status_code == 200:
                result.setdefault(self.saltMaster, {}).setdefault(minion, {})["status"] = True
                result.setdefault(self.saltMaster, {}).setdefault(minion, {})["comment"] = response.json()["return"][0].get(minion, "")
            else:
                result.setdefault(self.saltMaster, {}).setdefault(minion, {})["status"] = False
                result.setdefault(self.saltMaster, {}).setdefault(minion, {})["comment"] = "Salt Cmd Executed error!!!"
        return result

    def checkToken(self):
        masToken = self.__readToken().get(self.saltMaster, {}) ##获取这个Master的token
        if not masToken.get('token'): ##如果token不存在 重新获取token
            self.getToken()
        else: ##如果token存在则进行ping测试
            now = self.__dateToday()  ##当前的时间
            expire = float(masToken.get('deadTime'))
            if expire - now < 600:  ##当token还有十分钟，  重新获取token
                self.getToken()
        return self

    def start(self):
        result = {}
        self.checkToken()  ##检查token是否可用
        saltPing = self.saltCmdPing()
        masToken = self.__readToken().get(self.saltMaster, {}).get('token', '')
        for min in self.saltMinionList:
            if not saltPing[self.saltMaster][min]['status']:
                result.setdefault(self.saltMaster, {})[min] = saltPing[self.saltMaster][min]
            else:
                saltCmd = self.saltCmdExecute(min, masToken)
                for (k, v) in saltCmd.items():
                    result.setdefault(k, {}).update(v)
        return result

if __name__ == '__main__':
    x = saltApi(saltMaster="172.16.70.231", saltMasterPort=9000, saltMinion="172.16.70.233,172.16.70.232", saltFunc = "cmd.run", saltCmd = "uname -r")
    print(x.start())
```