```
# grains管理对象属性

1.1.1 原生Grains
Grains是saltstack组件中非常重要的组件之一，它是salt提供的一个收集操作系统底层信息的一个接口，它可以用来收集操作系统的域名 IP 内核 系统类型等信息．其中的内容是相对静态的，当操作系统中的配置发生了改变或人为的修改了Grains中的配置，那么需要刷新Grains


1.1.2 自定义Grains：

Minion的Grains信息是Minion启动时采集汇报给Master的，在实际的环境中需要根据自己的业务情况去自定义一些Grains．

自定义Grains的方法：

1．通过minion配置文件定义

2．通过saltstack master上通过Grains模块来定义

3．通过Python脚本定义


二．grains常用命令

2.1.1 列出相关函数

[root@k8s-master1 ~]# salt '172.16.70.232'sys.list_functions  grains
172.16.70.232:
    - grains.append
    - grains.delkey
    - grains.delval
    - grains.equals
    - grains.fetch
    - grains.filter_by
    - grains.get
    - grains.get_or_set_hash
    - grains.has_value
    - grains.item
    - grains.items
    - grains.ls
    - grains.remove
    - grains.set
    - grains.setval
    - grains.setvals

2.1.2 查看目标主机上有哪些grains (部分输出结果如下)

[root@k8s-master1 ~]#   salt '172.16.70.232'  grains.ls
172.16.70.232:
    - SSDs
    - biosreleasedate
    - biosversion
    - cpu_flags
    - cpu_model
    - cpuarch
    - dev
    - disks
      ..
      ..
      ..


2.1.3. 列出所有的Grains数据
[root@k8s-master1 salt]# salt '*' grains.items
172.16.70.232:
    systemd:
        ----------
        features:
            +PAM +AUDIT +SELINUX +IMA -APPARMOR +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP +GCRYPT +GNUTLS +ACL +XZ +LZ4 -SECCOMP +BLKID +ELFUTILS +KMOD +IDN
        version:
            219
    uid:
        0
    username:
        root
    uuid:
        564dc749-8603-1de1-b76d-523caf6eb49a
    virtual:
        VMware
    webapp:
        - nginx
    zfs_feature_flags:
        False
    zfs_support:
        False
    zmqversion:
        4.1.4
        ...


[root@k8s-master1 ~]# salt '172.16.70.232' grains.item roles

172.16.70.232:
    ----------
    roles:
        - salt-minion
        - k8s-node1
        - prometheus
        - kibana

[root@k8s-master1 ~]# salt '172.16.70.232' grains.item  env
172.16.70.232:
    ----------
    env:
        develop

[root@k8s-master1 ~]# salt '172.16.70.232' grains.item  idc
172.16.70.232:
    ----------
    idc:
        szd


三．  定义Grains

3.1.1 通过minion配置文件定义grains
1) 修改配置文件/etc/salt/minion中，打开default_include：minion.d/*.conf

2) 在minion端的/etc/salt/minion.d/目录下新建并编辑conf文件

3) 创建base.conf 并添加内容
[root@k8s-node1 minion.d]# cat base.conf
例如：
grains:
  roles:
    - salt-minion
    - k8s-node1
```

```
    - prometheus
    - kibana
  idc: szd
  env: develop

##通过这种方式定义的grains不能通过 grains.delkey 进行删除与修改, 如有删除修改需求, 要改该配置文件, 并重启minion才可以生效


3.1.2 通过/etc/salt/grains文件定义
[root@k8s-node1 salt]# vim  /etc/salt/grains
例如:
roles:
  - webserver
  - memcache
deployment: datacenter4
cabinet: 13
cab_u: 14-15

##通过这种方式定义的Grains可以通过 grains.delkey  grains.set 进行动态的删除与修改


3.1.3 通过Grains模块定义Grains

1) 在目标主机上获取一个不存在Grains属性
[root@k8s-master1 ~]# salt '172.16.70.232' grains.item  timezone
172.16.70.232:
    ----------
    timezone:

2) 通过grains.append设置值
[root@k8s-master1 ~]# salt '172.16.70.232'  grains.append timezone 'Shanghai/Asia'
172.16.70.232:
    ----------
    timezone:
        - Shanghai/Asia


3) 重新获取刚刚设置的gains值(可见获取成功)
[root@k8s-master1 ~]# salt '172.16.70.232' grains.item  timezone
172.16.70.232:
    ----------
    timezone:
        - Shanghai/Asia


4) 同时设置多个grains的属性
[root@k8s-master1 ~]# salt '172.16.70.232' grains.setvals "{'a': 'apple', 'b': 'banana'}"
172.16.70.232:
    ----------
    a:
        apple
    b:
        banana

5) 同时获取多个grains属性
[root@k8s-master1 ~]# salt '172.16.70.232' grains.item a b c
172.16.70.232:
    ----------
    a:
        apple
    b:
        banana
    c:

6) 通过Saltstack master的grains模块所设置的属性, 可以在minion的 /etc/salt/grains 配置文件中查看到.
[root@k8s-node1 minion.d]# cat /etc/salt/grains
a: apple
b: banana
dev: terryliu
hosttype:
- online
jobs: develop
ops: avwang
ping:
- pong
teacher: liu
timezone:
- Shanghai/Asia

7) 查看某grains是否有值
[root@k8s-master1 master.d]# salt '172.16.70.232' grains.has_value book

8) 删除grains的值
[root@k8s-master1 master.d]# salt '172.16.70.232' grains.delkey book

3.1.4 在saltmaster上使用自定义的Python脚本获取Grains信息

1) 将自定的Python脚本保存在saltstack master 的 file_roots 下定义的 base文件路径中, 我的当前配置为:
[root@k8s-master1 _grains]# cat /etc/salt/master | grep -v ^$ | grep -v "#"
default_include: master.d/*.conf
interface: 172.16.70.231
file_roots:
   base:
     - /srv/salt/
pillar_roots:
  base:
    - /srv/salt/_pillar

2) 在 /srv/salt 路径下创建文件夹 _grains

3) 开始编写Python grains模块脚本
[root@k8s-master1 _grains]# cat /srv/salt/_grains/os_base.py
#!/usr/bin/python
import commands

def f1():
    ip_addr = commands.getoutput('hostname -i')
    kernal = commands.getoutput('uname -r')
    result = {'ipaddr': ip_addr, 'kernal': kernal}
    return result

if __name__ == '__main__':
    print f1()

4) 将脚本模块脚本推送到目标主机
[root@k8s-master1 _grains]# salt '172.16.70.232' saltutil.sync_grains
172.16.70.232:
    - grains.os_base

该脚本推送到了minion节点的 /var/cache/salt/minion/extmods/grains 路径下

[root@k8s-node1 grains]# pwd
/var/cache/salt/minion/extmods/grains

[root@k8s-node1 grains]# ls
os_base.py  os_base.pyc
```

```
5) 在minion主机上获取, 自定义的grians属性信息.
[root@k8s-master1 _grains]# salt '172.16.70.232' grains.item ipaddr kernal

172.16.70.232:
    ----------
    ipaddr:
        172.16.70.232
    kernal:
        3.10.0-327.el7.x86_64

#通过这种方式设置的Grains不可以被grains.delkey()删除...
```

四. 筛选主机

4.1.1 根据grains上定义的角色来筛选主机.

使用 -G 参数

```
-G, --grain         Instead of using shell globs to evaluate the target
                        use a grain value to identify targets, the syntax for
                        the target is the grain key followed by a
                        globexpression: "os:Arch*".
```

例如:

```
[root@k8s-master1 ~]# salt -G "roles:k8s-node1" test.ping   #筛选roles属性为k8s-node1的主机
172.16.70.232:
    True
```

4.1.2 使用 -C 参数进行复合匹配

```
    -C, --compound      The compound target option allows for multiple target
                        types to be evaluated, allowing for greater
                        granularity in target matching. The compound target is
                        space delimited, targets other than globs are preceded
                        with an identifier matching the specific targets
                        argument type: salt 'G@os:RedHat and webser* or
                        E@database.*'.
```

注: 筛选所有idc为GZA并且网络区域在DMZ的主机

```
[root@k8s-master1 salt]# salt -C 'G@idc:gza and G@network-zone:dmz' test.ping
172.16.70.233:
    True
```

五. salt-api管理grains

```
d = {"client": "local", "tgt": '172.16.70.232', "fun": "grains.items"} ##1. 查看所有的Grains
d = {"client": "local", "tgt": '172.16.70.232', "fun": "grains.item", "arg": ["uid", "username"]} ##查看指定的grains
d = {"client": "local", "tgt": '172.16.70.232', "fun": "grains.get", "arg": "myapp"} ##查看指定的grains
d = {"client": "local", "tgt": '172.16.70.232', "fun": "grains.delkey", "arg": "myapp"} ##删除指定的grains
d = {"client": "local", "tgt": '172.16.70.232', "fun": "grains.get", "arg": "myapp"} ##查看指定的grains
d = {"client": "local", "tgt": '172.16.70.232', "fun": "grains.set", "arg": 'myapp nginx'} ##设置指定的grains
d = {"client": "local", "tgt": '172.16.70.232', "fun": "grains.get", "arg": "myapp"} ##查看指定的grains
d = {"client": "local", "tgt": '172.16.70.232', "fun": "grains.setvals", "arg": "{'a':1, 'b': 2}"} ##设置指定的grains
d = {"client": "local", "tgt": '172.16.70.232', "fun": "grains.item", "arg": ["a", "b"]} ##查看指定的grains
d = {"client": "local", "tgt": '172.16.70.232', "fun": "saltutil.sync_grains"} ##同步本地自己编写的grains到目标主机
key = "ad88c4afc9998d3ca081a640b0a8caea43cd6ae7"
headers = {"Accept": "application/json", "X-Auth-Token": key}
x = requests.post("http://172.16.70.231:9000", headers=headers, data=d)
print x.status_code
print x.text
```

In [ ]: