States配置管理

一.什么是States
States是saltstack中的配置语言，日常进行配置管理时需要大量编写States文件，
比如我们要安装一个包，管理一个配置文件，并保证服务的正常运行这些场景时，我们就需要编写states文件，来描述和实现我们的功能. States配置文件一般使用 YAML语言来编写，也支持用Python来编写. 我们下面的例子都采用YAML.

1.1.1 查看minion支持的所有的states列表

```
[root@k8s-master1 _modules]# salt '172.16.70.232' sys.list_state_modules
172.16.70.232:
    - acl
    - alias
    - alternatives
    - apache
    - archive
       ...
       ...
```

因为模块较多，这里只是部分显示.

1.1.2 查看指定status支持的function(我们以file为例，通过file我们可以管理配置文件)

```
[root@k8s-master1 _modules]# salt '172.16.70.232' sys.list_state_functions file
172.16.70.232:
    - file.absent
    - file.accumulated
    - file.append
    - file.blockreplace
    - file.cached
    - file.comment
        ...
        ...
```

1.1.3 查看status下的某个函数的使用方法(以file.append为例)

```
[root@k8s-master1 _states]# salt '172.16.70.232' sys.state_doc file.append  查看state指定function用法
       …
```

二. 一个简单的例子了解status，使用states对minion上的DNS服务配置文件进行统一管理

2.1.1 修改saltmaster配置文件，将/srv/salt/_states目录加入base环境

```
cat /etc/salt/master

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

2.1.2 编写states配置文件

```
[root@k8s-master1 _states]# cat one.sls
dns:
  file.managed:
    - name: /etc/resolv.conf
    - source: salt://config/resolv.conf
    - user: root
    - group: root
    - mode: 644
```

2.1.3 开始配置管理

```
[root@k8s-master1 _states]# salt '172.16.70.232' state.sls one

172.16.70.232:
----------
          ID: dns
    Function: file.managed
        Name: /etc/resolv.conf
      Result: True
     Comment: File /etc/resolv.conf updated
     Started: 17:24:16.016929
    Duration: 43.486 ms
     Changes:
              ----------
              diff:
                  ---
                  +++
                  @@ -5,3 +5,4 @@
                   ##33333333sdfsdf
                   ##this is new
                   #this is new 2
                  +#xxxyyzz

Summary for 172.16.70.232
------------
Succeeded: 1 (changed=1)
Failed:    0
------------
Total states run:     1
Total run time:  43.486 ms
```

总结：以上就完成了对minion主机的简单文件管理

2.2.1 在目标主机上部署JDK，编写State配置文件

```
[root@k8s-master1 _states]# cat jdk_install.sls
jdk_home:
  file.directory:
    - name: /usr/local
    - user: root
    - group: root
    - mode: 755


jdk_install:
  file.managed:
    - name: /usr/local/jdk-8u171-linux-x64.tar.gz
    - source: salt://jdk-8u171-linux-x64.tar.gz
    - mode: 755
    - user: root
    - group: root
  cmd.run:
    - name: cd /usr/local && tar -zxvf jdk-8u171-linux-x64.tar.gz

/etc/profile:
  file.append:
     - text:
        - export JAVA_HOME=/usr/local/jdk1.8.0_171
        - export PATH=$JAVA_HOME/bin:$PATH
  cmd.run:
    - name: source /etc/profile
```

2.2.2 开始部署

```
[root@k8s-master1 _states]# salt '172.16.70.232' state.sls  jdk_install
```

2.2.3 同时管理多台主机,minion1 minion2 都有配置文件/tmp/common.conf, 并且两台主机都有自己特殊的配置文件/tmp/minion1.conf /tmp/minion2.conf, 此时我们创建top.sls同时对多台机器进行配置管理.

```
[root@localhost _state]# cat top.sls
base:
  '*':
    - one
  '172.16.70.231':
    - two
  '172.16.70.232':
    - three

[root@localhost _state]# cat one.sls
one:
  file.managed:
    - name: /tmp/one.conf
    - source: salt://config/one.conf
    - user: root
    - group: root
    - mode: 644

[root@localhost _state]# cat two.sls
two:
  file.managed:
    - name: /tmp/two.conf
    - source: salt://config/two.conf
    - user: root
    - group: root
    - mode: 644

[root@localhost _state]# cat three.sls
three:
  file.managed:
    - name: /tmp/three.conf
    - source: salt://config/three.conf
    - user: root
    - group: root
    - mode: 644


[root@localhost _state]# salt '*' state.highstate 开始执行 highstate
```

2.2.4 state与pillar grains相结合配置目标主机, 假设有如下需求:

分别为业务nginx业务和mysql业务服务器创建账号 nginx 与 mysql

1) 通过Grains为主机打标签, 并查看配置.

```
[root@localhost _state]# salt '172.16.70.231' grains.setvals "{"service": "nginx"}"
[root@localhost _state]# salt '172.16.70.232' grains.setvals "{"service": "mysql"}"
```

查看主机grains的service配置

```
[root@localhost _state]# salt '172.16.70.231' grains.get service
172.16.70.231:
    nginx
[root@localhost _state]# salt '172.16.70.232' grains.get service
172.16.70.232:
    mysql
```

2) 定义pillar配置, 用来创建账号.

```
[root@localhost _state]# cat ../_pillar/useradd.sls
useradd:
    {% if grains['service'] == 'nginx' %}
    cmd: useradd nginx; echo 'nginx' | passwd --stdin  nginx
    {% elif grains['service'] == 'mysql' %}
    cmd: useradd mysql; echo 'mysql' | passwd --stdin  mysql
    {% endif %}
```

3) 验证pillar数据

```
[root@localhost _state]#  salt '*' pillar.items
172.16.70.232:
    ----------
    useradd:
        ----------
        cmd:
            useradd mysql; echo 'mysql' | passwd --stdin  mysql
    zabbix:
        ----------
        package-name:
            zabbix
        port:
            10050
        user:
            admin
        version:
            2.2.4
```

4) 创建state配置文件.

```
[root@localhost _state]# cat useradd.sls
useradd:
  cmd.run:
    - name: {{ pillar['useradd']['cmd'] }}
```

5) 开始创建账号

```
[root@localhost _state]# salt '*' state.sls useradd
```

6) 分别在两台机器上验证账号的创建.

```
[root@harbor-a salt]# id nginx
uid=996(nginx) gid=992(nginx) groups=992(nginx)
[root@harbor-a salt]# id mysql
id: mysql: no such user
[root@harbor-a salt]#

[root@harbor-b ~]# id nginx
id: nginx: no such user
[root@harbor-b ~]# id mysql
uid=1000(mysql) gid=1000(mysql) groups=1000(mysql)
[root@harbor-b ~]#
```