**今日学习目标:**

1. 能够安装saltstack服务器和客户端,并连接
2. 能够使用sls文件实现基本文件操作(文件创建,删除，修改)
3. 能够使用sls文件实现基本目录操作(目录创建,删除，修改权限)
4. 能够使用sls文件实现基本用户管理(用户创建,删除)
5. 能够使用sls文件实现基本组管理(用户组创建,删除)
6. 能够使用sls文件实现基本cron时间任务管理
7. 能够使用sls文件安装rpm软件包和启动服务
8. 能够使用sls文件远程执行脚本

# saltstack介绍

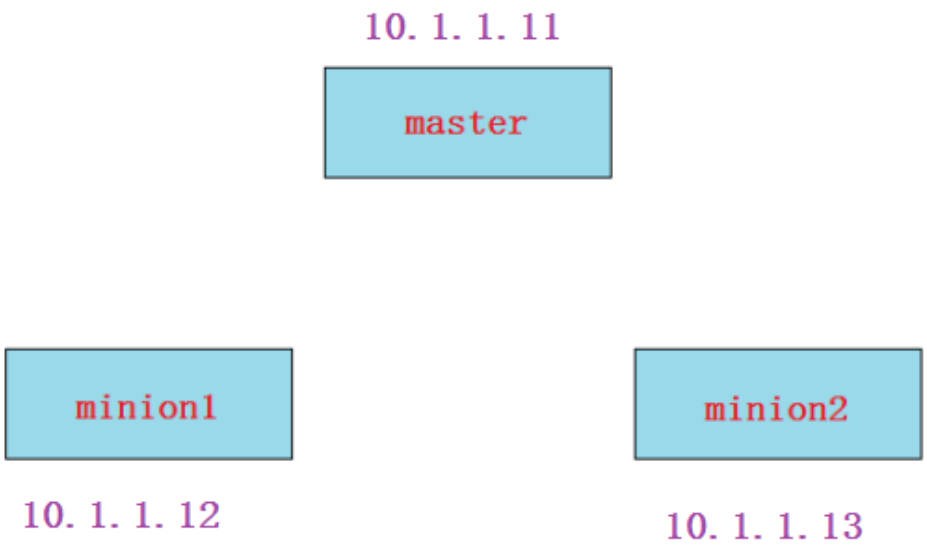
saltstack不是像ansible那样底层用ssh协议，它使用的基于ZeroMQ(一种轻量级的消息队列)来进行高效的网络通信。

官网的中文介绍:

http://docs.saltstack.cn/topics/index.html

# salt stack环境准备

**环境准备:**




1. 主机名及主机名绑定(**这个软件对主机名非常敏感，一定要先配置好主机名,然后绑定好**)

```
# vim /etc/hosts
10.1.1.11   master
10.1.1.12   minion1
10.1.1.13   minion2      这一次我都用短主机名(你用其它的也行，但要求确定并绑定后，就不要再乱改了)
```

2. 静态ip
3. 关闭防火墙,selinux
4. 时间同步
5. 安装epel源

```
master和所有minion服务器都要安装
# yum install epel-release
```

1. 安装软件包，定义主节点，启动服务

```
管理服务器上安装salt-master包，被管理节点安装salt-minion包（我这里master上也可以管理自己，所以master上也安装了salt-minion包）
[root@master ~]# yum install salt-master salt-minion
[root@minion1 ~]# yum install salt-minion
[root@minion2 ~]# yum install salt-minion
```

2. 所有被管理节点上指定master的ip

```
[root@master ~]# vim /etc/salt/minion
16 master: 10.1.1.11
[root@minion1 ~]# vim /etc/salt/minion
16 master: 10.1.1.11
[root@minion2 ~]# vim /etc/salt/minion
16 master: 10.1.1.11
```

3. 各节点启动相应的服务,并设置为开机自动启动

```
[root@master ~]# systemctl start salt-master
[root@master ~]# systemctl enable salt-master
[root@master ~]# systemctl start salt-minion
[root@master ~]# systemctl enable salt-minion

[root@minion1 ~]# systemctl start salt-minion
[root@minion1 ~]# systemctl enable salt-minion
[root@minion2 ~]# systemctl start salt-minion
[root@minion2 ~]# systemctl enable salt-minion
```

4. 建立连接

```
[root@master ~]# salt-key           --使用此命令查看有哪些Unaccepted Keys
Accepted Keys:
Denied Keys:
Unaccepted Keys:
master
```

```
minion1
minion2
Rejected Keys:


使用-a参数表示接受，把Unaccepted Keys变成accept状态（也可以使用salt-key -A命令一次性全接受）
[root@master ~]# salt-key -a minion1
[root@master ~]# salt-key -a minion2
[root@master ~]# salt-key -a master

[root@master ~]# salt-key   --再次查看验证
Accepted Keys:
master
minion1
minion2
Denied Keys:
Unaccepted Keys:
Rejected Keys:
```

5. 测试连接

```
[root@master ~]# salt '*' test.ping
minion1:
    True
minion2:
    True
master:
    True

[root@master ~]# salt-run manage.up         --查看up状态的被管理机器
- master
- minion1
- minion2
```

# 远程命令操作

你可以通过cmd.run来传linux的命令，和ansible里的shell模块类似.

```
*号代表所有accepted keys的机器。
还可以使用以下方式来匹配
# salt "minion*"
# salt "minion[12]"
# salt -E "minion(1|2)"
# salt -L "minion1,minion2"
[root@master ~]# salt "*" cmd.run "uname -r"
master:
    3.10.0-514.el7.x86_64
minion2:
    3.10.0-514.el7.x86_64
minion1:
    3.10.0-514.el7.x86_64
```

# grains

> 问题: 如果我管理的多台机器中有各种不同的OS平台，我现在想针对所有的centos平台创建一个文件，那么如何得知哪些是centos平台的机器呢?
>
> 答: 使用grains收集信息，然后匹配，实现服务器分组。

grains用于收集被管理主机的基本信息，包括操作系统平台，内核版本等。在服务器端可以根据这些信息进行灵活定制，或者匹配对应的机器。

#### 通过收集的信息匹配对应的机器

```
获取minion1的所有items名称
[root@master ~]# salt "minion1" grains.ls
获取minion1的所有items名称及值
[root@master ~]# salt "minion1" grains.items

通过grains获取的os平台信息来匹配操作对应的机器（这里要用-G参数）
[root@master ~]# salt -G os:CentOS cmd.run 'hostname'
master:
    master
minion1:
    minion1
minion2:
    minion2
```

#### 通过grains定义客户端服务器角色(role)

```
此配置文件最后加上下面一段，表示minion1属于webserver角色也属于dbserver角色
[root@minion1 ~]# vim /etc/salt/minion
grains:
  roles:
```

```
    - webserver
    - dbserver

改了配置文件，重启服务生效
[root@minion1 ~]# systemctl restart salt-minion

[root@minion2 ~]# vim /etc/salt/minion
grains:
  role:
    - webserver
    - dbserver

[root@minion2 ~]# systemctl restart salt-minion
```

master上通过roles:webserver来操作对应的被管理机

```
[root@master ~]# salt -G roles:webserver cmd.run 'hostname'
minion1:
    minion1
minion2:
    minion2
```

master上通过roles:dbserver来操作对应的被管理机

```
[root@master ~]# salt -G roles:dbserver cmd.run 'hostname'
minion1:
    minion1
minion2:
    minion2
```

# salt states管理

salt states(SLS)文件是用来管理saltstack的核心，由它来进行配置管理。

参考文档

http://docs.saltstack.cn/topics/tutorials/starting_states.html

module文档

http://docs.saltstack.cn/ref/index.html

**定义sls文件的根目录**

```
[root@master ~]# vim /etc/salt/master       --找到下面几句，并打开注释

file_roots:
   base:                    --base是测试环境，prod是生产环境
     - /srv/salt/

[root@master ~]# systemctl restart salt-master
[root@master ~]# mkdir /srv/salt
```

# 文件操作

参考http://docs.saltstack.cn/ref/states/all/salt.states.file.html#module-salt.states.file

### 定义top.sls文件

```
写这种文件(yaml格式)不要使用tab键，都使用空格，否则会报Illegal tab character错误
[root@master ~]# vim /srv/salt/top.sls      # top.sls是所有配置管理文件的入口文件
base:                    # 代表base环境
 "minion*":              # minion*代表所有minion开头的节点(表示运行top.sls只针对匹配的节点生效)
 - file                  # 对应下面的/srv/salt/file.sls文件
```

```
[root@master ~]# vim /srv/salt/top.sls
base:
 "os:CentOS":
 - match: grain
 - file
```

### 创建文件

(**注意:如果执行时报字符相关的错误，请手动输入，不要复制我的.复制markdown的没问题，复制pdf文档会有字符格式问题**)

```
[root@master ~]# vim /srv/salt/file.sls
/tmp/1.txt:
  file.managed:
    - contents: haha
```

使用**state.sls**指令执行特定sls文件

```
[root@master ~]# salt "minion*" state.sls file
```

### 删除文件

```
[root@master ~]# vim /srv/salt/file.sls
sdfdsafsa:
  file.absent:
      - name: /tmp/1.txt
[root@master ~]# salt "minion*" state.sls file
```

**拷贝文件并修改属性**

```
[root@master ~]# vim /srv/salt/file.sls
/tmp/2.txt:
  file.managed:
      - source: /etc/fstab
      - user: bin
      - group: daemon
      - mode: 0600

[root@master ~]# salt "minion*" state.sls file
```

# 目录操作

参考http://docs.saltstack.cn/ref/states/all/salt.states.file.html#module-salt.states.file

**创建目录并修改属性**

```
[root@master ~]# vim /srv/salt/top.sls
base:
 "minion*":
  - file
  - dir                 --加上dir

[root@master ~]# vim /srv/salt/dir.sls
/tmp/dir1:
  file.directory:
    - user: daemon
    - group: daemon
    - dir_mode: 0700

[root@master ~]# salt "minion*" state.sls dir
```

扩展: state.sls与state.highstate的区别

```
下面这条命令执行时是直接读取/srv/salt/dir.sls文件,所以在top.sls文件里是否加- dir不影响
[root@master ~]# salt "minion*" state.sls dir
```

```
下面这条命令执行时是直接读取/srv/salt/top.sls文件,所以在top.sls文件里必须加- dir,否则就只能读/srv/salt/file.sls，而不能读/srv/salt/dir.sls文件了

[root@master ~]# salt "*" state.highstate
```

为了方便: 以下笔记我全部使用state.sls指令，不再加到top.sls文件中了

**删除目录**

```
[root@master ~]# vim /srv/salt/dir.sls
/tmp/dir1:
  file.absent:
    - name: /tmp/dir1
[root@master ~]# salt "minion*" state.sls dir
```

**把master上的目录同步到远程**

```
[root@master ~]# vim /srv/salt/dir.sls
delete dir:
 file.absent:
   - name: /tmp/haha_dir

rsync dir:
  file.recurse:
    - name: /tmp/haha_dir
    - source: salt://test_dir
    - user: root
    - dir_mode: 0700
    - file_mode: 0600
    - mkdir: True
    - clean: True

[root@master ~]# mkdir /srv/salt/test_dir   --对应上面的- source: salt://test_dir配置。
[root@master ~]# echo haha > /srv/salt/test_dir/1.txt

执行成功后，会把/srv/salt/test_dir目录里的同步到minion1和minion2的/tmp/haha_dir,并改相应的权限
[root@master ~]# salt "minion*" state.sls dir
```

**扩展:直接命令拷贝文件或目录**

```
把master上base目录里的test_dir/1.sh拷到客户机上为/tmp/2.sh
[root@master ~]# salt "minion*" cp.get_file salt://test_dir/1.sh /tmp/2.sh
minion1:
    /tmp/2.sh
minion2:
    /tmp/2.sh
```

```
把master上base目录里的test_dir目录全拷到客户机上的/tmp/目录下
[root@master ~]# salt "minion*" cp.get_dir salt://test_dir/ /tmp/
minion2:
    - /tmp//test_dir/1.sh
    - /tmp//test_dir/1.txt
minion1:
    - /tmp//test_dir/1.sh
    - /tmp//test_dir/1.txt
```

# 用户管理

参考http://docs.saltstack.cn/ref/states/all/salt.states.user.html#module-salt.states.user

**创建用户usera**

```
[root@master ~]# vim /srv/salt/user.sls
usera:
  user.present:
   - name: usera
```

**创建用户userb,删除usera**

```
如果没有5000gid的组，就把- gid: 5000这一句去掉，否则创建不成功
[root@master ~]# vim /srv/salt/user.sls
userb:
  user.present:
    - fullname: daniel
    - shell: /bin/bash
    - home: /home/userb
    - uid: 5000
    - gid: 5000
    - groups:
        - root
        - bin
        - daemon

usera:
  user.absent:
         - purge: true

 - purge: true表示删除用户的同时删除用户家目录，邮件等数据
[root@master ~]# salt "minion*" state.sls user
```

# 组管理

参考http://docs.saltstack.cn/ref/states/all/salt.states.group.html#module-salt.states.group

**组的创建**

```
[root@master ~]# vim /srv/salt/group.sls
group1:
 group.present:
    - gid: 6002
    - addusers:
      - bin
      - daemon

group2:
 group.present:
   - gid: 7000
   - members:
     - root
     - lp

[root@master ~]# salt "minion*" state.sls group
```

**组的删除**

```
[root@master ~]# vim /srv/salt/group.sls
group1:
 group.absent:
  - name: group1

group2:
 group.absent:
  - name: group2

[root@master ~]# salt "minion*" state.sls group
```

# 时间任务管理

参考http://docs.saltstack.cn/ref/states/all/salt.states.cron.html#module-salt.states.cron

**增加时间任务**

```
[root@master ~]# vim /srv/salt/cron.sls
cron_jobs:
  cron.present:
  - name: touch /tmp/cron1
  - user: root
  - minute: '*/2'
  - hour: 16

[root@master ~]# salt "minion*" state.sls cron
```

**删除时间任务**

```
[root@master ~]# vim /srv/salt/cron.sls
cron_jobs:
 cron.absent:
  - name: touch /tmp/cron1

[root@master ~]# salt "minion*" state.sls cron
```

## 软件包与服务管理

参考http://docs.saltstack.cn/ref/states/all/salt.states.pkg.html#module-salt.states.pkg

参考http://docs.saltstack.cn/ref/states/all/salt.states.service.html#module-salt.states.service

```
[root@master ~]# vim /srv/salt/httpd.sls
httpd-install:
  pkg.installed:
    - names:
      - httpd
      - httpd-devel

httpd-config:
  file.managed:
    - name: /etc/httpd/conf/httpd.conf
    - source: salt://test_dir/httpd.conf  --在master的/srv/slat/test_dir/目录里准备一个已经配置好的httpd.conf配置文件

httpd-run:
  service.running:
      - name: httpd
      - enable: True

[root@master ~]# salt "minion*" state.sls httpd
```

## 远程执行脚本

```
[root@master ~]# vim /srv/salt/test_dir/1.sh        --在master上的base目录准备好脚本
#!/bin/bash

mkdir /test -p
touch /test/{1..10}

[root@master ~]# vim /srv/salt/script.sls
shell_test:
 cmd.script:
   - source: salt://test_dir/1.sh  --对应master上准备好的脚本
   - user: root

[root@master ~]# salt "minion*" state.sls script
```

```
扩展:直接命令执行shell脚本
[root@master ~]# salt "minion*" cmd.script salt://test_dir/1.sh          --执行base目录里的test_dir/1.sh脚本（命令在master上操作，但脚本却是在被管理机器上执行的）
```

# 综合练习

**使用sls文件搭建rpm版的lnmp**

```
[root@master ~]# mkdir /srv/salt/lnmp/conf  -p      --然后在此目录里准备好lnmp需要的配置文件（过程省略）

[root@master ~]# vim /srv/salt/lnmp.sls
lnmp-install:
  pkg.installed:
    - names:
      - mariadb
      - mariadb-server
      - php
      - php-mysql
      - php-gd
      - libjpeg-turbo-devel
      - php-ldap
      - php-odbc
      - php-pear
      - php-xml
      - php-xmlrpc
      - php-mbstring
      - php-bcmath
      - php-common
      - php-fpm
      - php-pecl-zendopcache
      - nginx
      - memcached
```

```
      - php-pecl-memcache

php-fpm-config:
  file.managed:
    - name: /etc/php-fpm.d/www.conf
    - source: salt://lnmp/conf/www.conf     --提前准备好php-fpm配置文件

nginx-config:
  file.managed:
    - name: /etc/nginx/nginx.conf
    - source: salt://lnmp/conf/nginx.conf   --提前准备好nginx配置文件

test.php:
  file.managed:
    - name: /usr/share/nginx/html/test.php
    - source: salt://lnmp/conf/test.php     --提前准备好php测试文件

/var/run/php-fpm/:
  file.directory:
    - user: nginx
    - group: nginx

mariadb-run:
  service.running:
      - name: mariadb
      - enable: True

php-fpm-run:
  service.running:
      - name: php-fpm
      - enable: True

nginx-run:
  service.running:
      - name: nginx
      - enable: True

memcached-run:
  service.running:
      - name: memcached
      - enable: True

[root@master ~]# salt "minion*" state.sls lnmp
```