环境介绍

OS版本：

CentOS Linux release 7.8.2003

硬件最低配置：

2 CPU、 4G 内存 、20G 磁盘

主机名(FQDN)	IP 地址(NAT)	角色
linux-node1.example.com	eth0:192.168.56.11	salt-master 和 salt-minion
linux-node2.example.com	eth0:192.168.56.12	salt-minion

注意：设置 DNS 解析或修改 hosts 文件，实现 ping 主机名或者 FQDN 得到正确解析。

系统初始化

必要设置

注：以下配置操作，每台均需要执行。

```
# 更新 CentOS 系统
sudo yum update && yum upgrade
# 安装 salt 源
sudo yum install -y https://repo.saltstack.com/py3/redhat/salt-py3-repo-
latest.el7.noarch.rpm
sudo yum clean expire-cache
# 关闭防火墙
systemctl stop firewalld
# 永久禁用防火墙
systemctl disable firewalld
```

设置主机名

```
[root@localhost ~]# vi /etc/hostname
// 根据每台机器实际名称进行修改
linux-node1.example.com
```

设置 IP 地址

```
[root@localhost ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
# 根据每台机器实际信息进行修改
TYPE=Ethernet
NAME=ens33
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.56.11
NETMASK=255.255.255.0
GATEWAY=192.168.56.1
DNS1=114.114.114.114
DNS2=8.8.8.8
DOMAIN="example.com"
```

设置主机名解析

```
[root@localhost ~]# vi /etc/hosts

127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.56.11 linux-node1 linux-node1.example.com
192.168.56.12 linux-node2 linux-node2.example.com
```

重启网络

```
systemctl restart network
```

安装 salt

安装 salt-master

ssh 登录到 `linux-node1.example.com` 节点进行如下命令：

```
# 安装 salt-master salt-minion salt-ssh
sudo yum install -y salt-master salt-minion salt-ssh
# 启动 salt-master 服务
sudo systemctl enable salt-master
sudo systemctl start salt-master
# 查看 salt-master 服务
sudo systemctl status salt-master
```

查看 salt-master 包安装文件

```
[root@linux-node1 ~]# rpm -ql salt-master
/etc/salt/master
/etc/salt/master.d
/etc/salt/pki/master
/usr/bin/salt
/usr/bin/salt-cp
/usr/bin/salt-key
/usr/bin/salt-master
/usr/bin/salt-run
/usr/bin/salt-unity
/usr/lib/systemd/system/salt-master.service
/usr/share/man/man1/salt-cp.1.gz
/usr/share/man/man1/salt-key.1.gz
/usr/share/man/man1/salt-master.1.gz
/usr/share/man/man1/salt-run.1.gz
/usr/share/man/man1/salt-unity.1.gz
/usr/share/man/man1/salt.1.gz
/usr/share/man/man7/salt.7.gz
```

安装 salt-minion

ssh 登录到 `linux-node2.example.com` 节点进行如下命令：

```
sudo yum install -y salt-minion
```

配置 minion 连接

ssh 登陆`linux-node2.example.com`编辑 `/etc/salt/minion` 文件中 `master` 地址和显示`id`

```
[root@linux-node2 ~]# vim /etc/salt/minion
##### Primary configuration settings #####
```

```
##########################################
# This configuration file is used to manage the behavior of the Salt Minion.
# With the exception of the location of the Salt Master Server, values that are
# commented out but have an empty line after the comment are defaults that need
# not be set in the config. If there is no blank line after the comment, the
# value is presented as an example and is not the default.

# Per default the minion will automatically include all config files
# from minion.d/*.conf (minion.d is a directory in the same directory
# 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
# resolved, then the minion will fail to start.
#master: salt
//修改为 master 的 FQDN 或者 IP 地址
master: linux-node1.example.com

# Explicitly declare the id for this minion to use, if left commented the id
# will be the hostname as returned by the python call: socket.getfqdn()
# Since salt uses detached ids it is possible to run multiple minions on the
# same machine but with different ids, this can be useful for salt compute
# clusters.
#id:
//修改为 minion 显示信息，默认 salt 会调用该节点的 FQDN 作为 id
id: linux-node2.example.com
```

查看 `pki` 证书信息

```
[root@linux-node2 ~]# tree /etc/salt/
/etc/salt
├── cloud
├── cloud.conf.d
├── cloud.deploy.d
├── cloud.maps.d
├── cloud.profiles.d
├── cloud.providers.d
├── master
├── master.d
├── minion
├── minion.d
├── pki
│   ├── master
│   └── minion
├── proxy
├── proxy.d
└── roster
```

启动 salt-minion 服务

```
[root@linux-node2 ~]# sudo systemctl enable salt-minion
[root@linux-node2 ~]# sudo systemctl start salt-minion
[root@linux-node2 ~]# tree /etc/salt/
/etc/salt
├── cloud
├── cloud.conf.d
├── cloud.deploy.d
├── cloud.maps.d
├── cloud.profiles.d
├── cloud.providers.d
├── master
├── master.d
├── minion
├── minion.d
├── pki
│   ├── master
│   └── minion
│       ├── minion.pem //私钥文件
│       └── minion.pub //公钥文件
├── proxy
├── proxy.d
└── roster

# 查看 salt-minion 服务
sudo systemctl status salt-minion
// 显示结果:
● salt-minion.service - The Salt Minion
   Loaded: loaded (/usr/lib/systemd/system/salt-minion.service; enabled; vendor
preset: disabled)
   Active: active (running) since 2020-12-17 16:58:26 HKT; 1min 10s ago
     Docs: man:salt-minion(1)
           file:///usr/share/doc/salt/html/contents.html
           https://docs.saltstack.com/en/latest/contents.html
 Main PID: 11443 (salt-minion)
   CGroup: /system.slice/salt-minion.service
           ├─11443 /usr/bin/python3 /usr/bin/salt-minion
           ├─11448 /usr/bin/python3 /usr/bin/salt-minion
           └─11450 /usr/bin/python3 /usr/bin/salt-minion

12月 17 16:58:26 linux-node2.example.com  systemd[1]: Starting The Salt
Minion...
12月 17 16:58:26 linux-node2.example.com  systemd[1]: Started The Salt Minion.
12月 17 16:58:37 linux-node2.example.com  salt-minion[11443]: [ERROR   ] The
Salt Master has cach...e
12月 17 16:58:47 linux-node2.example.com  salt-minion[11443]: [ERROR   ] The
Salt Master has cach...e
```

```
Hint: Some lines were ellipsized, use -l to show in full.
```

注： `linux-node1.example.com` 操作方式相同。

master key 管理

首次 `minion` 与 `master` 通信时 `minion` 会将自己创建的公钥(minion.pub)发送给 `master` ,在 `master` 端显示为 id 名称。ssh 登录到 `linux-node1.example.com` 节点进行如下命令：

```
[root@linux-node1 ~]# tree /etc/salt/
/etc/salt/
├── cloud
├── cloud.conf.d
├── cloud.deploy.d
├── cloud.maps.d
├── cloud.profiles.d
├── cloud.providers.d
├── master
├── master.d
├── minion
├── minion.d
├── minion_id
├── pki
│   ├── master
│   │   ├── master.pem
│   │   ├── master.pub
│   │   ├── minions
│   │   ├── minions_autosign
│   │   ├── minions_denied
│   │   ├── minions_pre //等待接受文件夹
│   │   │   ├── linux-node1.example.com  //linux-node1 的公钥文件
│   │   │   └── linux-node2.example.com  //linux-node2 的公钥文件
│   │   └── minions_rejected
│   └── minion
│       ├── minion.pem //私钥文件
│       └── minion.pub //公钥文件
├── proxy
├── proxy.d
└── roster
# 查询 key 证书
[root@linux-node1 ~]# salt-key -L
Accepted Keys:
Denied Keys:
Unaccepted Keys:
linux-node1.example.com
linux-node2.example.com
```

```
# 接受所有 key 证书
[root@linux-node1 ~]# salt-key -A
The following keys are going to be accepted:
Unaccepted Keys:
linux-node1.example.com
linux-node2.example.com
//输入y
Proceed? [n/Y] y
Key for minion linux-node1.example.com accepted.
Key for minion linux-node2.example.com accepted.

# 查询 key 证书
[root@linux-node1 ~]# salt-key -L
Accepted Keys:
linux-node1.example.com
linux-node2.example.com
Denied Keys:
Unaccepted Keys:
Rejected Keys:

# 查询证书文件变化
[root@linux-node1 ~]# tree /etc/salt/
/etc/salt/
├── cloud
├── cloud.conf.d
├── cloud.deploy.d
├── cloud.maps.d
├── cloud.profiles.d
├── cloud.providers.d
├── master
├── master.d
├── minion
├── minion.d
│   └── _schedule.conf
├── minion_id
├── pki
│   ├── master
│   │   ├── master.pem
│   │   ├── master.pub
│   │   ├── minions //已接受文件夹
│   │   │   ├── linux-node1.example.com  //已接受 linux-node1 的公钥文件
│   │   │   └── linux-node2.example.com  //已接受 linux-node2 的公钥文件
│   │   ├── minions_autosign
│   │   ├── minions_denied
│   │   ├── minions_pre
│   │   └── minions_rejected
│   └── minion
│       ├── minion_master.pub // master 公钥文件
│       ├── minion.pem
```

```
│       └── minion.pub
├── proxy
├── proxy.d
└── roster
```

salt 管理

常用命令

ssh 登录到 `linux-node1.example.com` 节点进行如下操作：

远程执行模块

```
[root@linux-node1 ~]# salt '*' test.ping
linux-node1.example.com:
    True
linux-node2.example.com:
    True
```

命令格式说明：`salt '*' test.ping`

`salt` 为执行命令， `*` 代表执行目标，`*`指所有节点，由于 `*` 在命令里面有特殊的含义，所以需要增加双引号或者单引号， `test.ping` 调用的执行模块

模块默认存储路径：`/usr/lib/python3.6/site-packages/salt/modules/`

`test.ping` 为该路径下的 `test.py` 文件

远程执行模块查询：https://docs.saltstack.com/en/latest/ref/modules/all/index.html

执行测试命令

```
[root@linux-node1 ~]# salt '*' cmd.run 'df -h'
linux-node1.example.com:
    Filesystem               Size  Used Avail Use% Mounted on
    devtmpfs                 899M     0  899M   0% /dev
    tmpfs                    910M  240K  910M   1% /dev/shm
    tmpfs                    910M  9.6M  901M   2% /run
    tmpfs                    910M     0  910M   0% /sys/fs/cgroup
    /dev/mapper/centos-root   39G  2.6G   36G   7% /
    /dev/sda1               1014M  321M  694M  32% /boot
    /dev/mapper/centos-home   19G   33M   19G   1% /home
    tmpfs                    182M     0  182M   0% /run/user/0
linux-node2.example.com:
    Filesystem               Size  Used Avail Use% Mounted on
    devtmpfs                 899M     0  899M   0% /dev
```

```
    tmpfs                    910M   40K  910M   1% /dev/shm
    tmpfs                    910M  9.6M  901M   2% /run
    tmpfs                    910M     0  910M   0% /sys/fs/cgroup
    /dev/mapper/centos-root   39G  2.6G   36G   7% /
    /dev/sda1               1014M  321M  694M  32% /boot
    /dev/mapper/centos-home   19G   33M   19G   1% /home
    tmpfs                    182M     0  182M   0% /run/user/0
```

命令格式说明：`salt '*' cmd.run 'df -h'`

`salt` 为执行命令， `*` 代表执行目标，`*`指所有节点，由于 `*` 在命令里面有特殊的含义，所以需要增加双引号或者单引号， `cmd.run` 调用的执行模块，`'df -h'` 所执行的命令

查看监听端口信息

```
[root@linux-node1 ~]# lsof -n -i:4505
COMMAND     PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
salt-mast  9776 root   18u  IPv4 100147      0t0  TCP *:4505 (LISTEN)
salt-mast  9776 root   20u  IPv4 122336      0t0  TCP 192.168.56.11:4505-
>192.168.56.11:60156 (ESTABLISHED)
salt-mast  9776 root   21u  IPv4 120865      0t0  TCP 192.168.56.11:4505-
>192.168.56.12:38448 (ESTABLISHED)
salt-mini 11397 root   22u  IPv4 122335      0t0  TCP 192.168.56.11:60156-
>192.168.56.11:4505 (ESTABLISHED)
```

`master` 服务启动后会监听 `4505` 端口用于 `minion` 的连接，发送相应指令

```
[root@linux-node1 ~]# lsof -n -i:4506
COMMAND    PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
salt-mast 9782 root   31u  IPv4 100985      0t0  TCP *:4506 (LISTEN)
```

`master` 服务启动后会监听 `4506` 端口用于接受 `minion` 的返回信息

配置管理 sls

Salt 通过状态实现配置管理，默认情况下通过 `yaml` 文件来描述一个状态，ssh 登录到 `linux-node1.example.com` 节点进行如下操作：

创建状态文件夹

```
[root@linux-node1 ~]# mkdir -p /srv/salt/{base,dev,test,prod}
[root@linux-node1 ~]# tree /srv/salt/
/srv/salt/
├── base
├── dev
├── prod
└── test

4 directories, 0 files
```

编辑 `master` 文件，设置其对应的文件夹

```
[root@linux-node1 ~]# vim /etc/salt/master
# Example:
# file_roots:
#   base:
#     - /srv/salt/
#   dev:
#     - /srv/salt/dev/services
#     - /srv/salt/dev/states
#   prod:
#     - /srv/salt/prod/services
#     - /srv/salt/prod/states
#
//增加如下信息
file_roots:
//2个空格
  base:
//4个空格 - 1个空格
    - /srv/salt/base
  dev:
    - /srv/salt/dev
  test:
    - /srv/salt/test
  prod:
    - /srv/saly/prod
```

重启 `master` 服务

```
[root@linux-node1 ~]# systemctl restart salt-master
```

创建分类的状态文件

```
[root@linux-node1 ~]# mkdir /srv/salt/base/web
[root@linux-node1 ~]# cd /srv/salt/base/web/
[root@linux-node1 web]# vim apache.sls
apache-install:
  pkg.installed:
    - name: httpd
apache-service:
  service.running:
    - name: httpd
    - enable: True
```

格式编写说明：https://docs.saltstack.com/en/getstarted/config/functions.html

状态模块信息：https://docs.saltstack.com/en/latest/ref/states/all/

执行 `sls` 文件

```
[root@linux-node1 web]# salt 'linux-node2*' state.sls web.apache
linux-node2.example.com:
----------
          ID: apache-install
    Function: pkg.installed
        Name: httpd
      Result: True
     Comment: The following packages were installed/updated: httpd
     Started: 23:57:19.091033
    Duration: 18767.875 ms
     Changes:
              ----------
              httpd:
                  ----------
                  new:
                      2.4.6-97.el7.centos
                  old:
              httpd-tools:
                  ----------
                  new:
                      2.4.6-97.el7.centos
                  old:
              mailcap:
                  ----------
                  new:
                      2.1.41-2.el7
                  old:
----------
          ID: apache-service
    Function: service.running
        Name: httpd
      Result: True
```

```
     Comment: Service httpd has been enabled, and is running
     Started: 23:57:37.870821
    Duration: 10464.805 ms
     Changes:
              ----------
              httpd:
                  True

Summary for linux-node2.example.com
------------
Succeeded: 2 (changed=2)
Failed:    0
------------
Total states run:     2
Total run time:  29.233 s
```

在这里面可以查询到2个状态过程改变，

1. ID: apache-install 安装一个 Apache
2. ID: apache-service 确保该服务为运行状态

这里面的执行时间与环境有关，例如 `yum` 执行快慢，ssh 登录到 `linux-node2.example.com` 节点执行：

```
# 查询 yum 执行状态
[root@linux-node2 ~]# ps aux | grep yum
root       2657  0.0  0.0 112828   976 pts/0    S+   23:57   0:00 grep --
color=auto yum
# 查询 httpd 服务状态
[root@linux-node2 ~]# systemctl status httpd
# 停止 httpd 服务
[root@linux-node2 ~]# systemctl stop httpd
```

再次在 `linux-node1.example.com` 执行 `sls` 文件

```
[root@linux-node1 web]# salt 'linux-node2*' state.sls web.apache

linux-node2.example.com:
----------

          ID: apache-install
    Function: pkg.installed
        Name: httpd
      Result: True
     Comment: All specified packages are already installed
     Started: 23:59:22.353893
    Duration: 1187.179 ms
```

```
     Changes:

----------

          ID: apache-service
    Function: service.running
        Name: httpd
      Result: True
     Comment: Service httpd is already enabled, and is running
     Started: 23:59:23.542390
    Duration: 10323.941 ms
     Changes:
              ----------
              httpd:
                  True

Summary for linux-node2.example.com
------------

Succeeded: 2 (changed=1)

Failed:    0
------------

Total states run:     2
Total run time:  11.511 s
```

在这里面可以查询到1个状态过程改变,

1. ID: apache-service 确保该服务为运行状态

设置 redis 状态

```
[root@linux-node1 web]# yum -y install redis
[root@linux-node1 web]# vim redis.sls
redis-install:
  pkg.installed:
    - name: redis
```

所有节点均执行 `sls` 文件

```
[root@linux-node1 web]# salt '*' state.sls web.redis

linux-node1.example.com:
----------

          ID: redis-install
```

```
    Function: pkg.installed
        Name: redis
      Result: True
     Comment: All specified packages are already installed
     Started: 00:14:33.369193
    Duration: 1194.225 ms
     Changes:

Summary for linux-node1.example.com
------------

Succeeded: 1

Failed:    0
------------

Total states run:     1
Total run time:   1.194 s

linux-node2.example.com:
----------

          ID: redis-install
    Function: pkg.installed
        Name: redis
      Result: True
     Comment: The following packages were installed/updated: redis
     Started: 00:14:33.116765
    Duration: 20867.543 ms
     Changes:
              ----------
              jemalloc:
                  ----------
                  new:
                      3.6.0-1.el7
                  old:
              redis:
                  ----------
                  new:
                      3.2.12-2.el7
                  old:

Summary for linux-node2.example.com
------------

Succeeded: 1 (changed=1)

Failed:    0
------------
```

```
Total states run:     1
Total run time:  20.868 s
```

由于我们提前在 `linux-node1.example.com` 节点上执行了 `yum -y install redis`，所有本次 `linux-node1.example.com` 并未发生改变。

设置高级状态，简单理解为一个执行的合集，定义那些节点应该执行那些 `sls` 文件

```
[root@linux-node1 web]# cd /srv/salt/base/
[root@linux-node1 base]# vim top.sls
base:
  'linux-node1.example.com':
    - web.redis
  'linux-node2.example.com':
    - web.apache
    - web.redis
```

执行高级状态

```
[root@linux-node1 base]# salt '*' state.highstate

linux-node1.example.com:
----------

          ID: redis-install
    Function: pkg.installed
        Name: redis
      Result: True
     Comment: All specified packages are already installed
     Started: 00:28:45.346077
    Duration: 1202.651 ms
     Changes:

Summary for linux-node1.example.com
------------

Succeeded: 1

Failed:    0
------------

Total states run:     1
Total run time:   1.203 s

linux-node2.example.com:
----------
```

```
          ID: apache-install
    Function: pkg.installed
        Name: httpd
      Result: True
     Comment: All specified packages are already installed
     Started: 00:28:45.421837
    Duration: 1200.206 ms
     Changes:

----------

          ID: apache-service
    Function: service.running
        Name: httpd
      Result: True
     Comment: The service httpd is already running
     Started: 00:28:46.623365
    Duration: 37.689 ms
     Changes:

----------

          ID: redis-install
    Function: pkg.installed
        Name: redis
      Result: True
     Comment: All specified packages are already installed
     Started: 00:28:46.661394
    Duration: 23.746 ms
     Changes:

Summary for linux-node2.example.com
------------

Succeeded: 3

Failed:    0
------------

Total states run:     3
Total run time:   1.262 s
```

salt ssh

`linux-node1.example.com` 和 `linux-node2.example.com` 节点进行如下操作：

```
yum install -y salt-ssh
```

ssh 登录到 `linux-node1.example.com` 节点进行如下操作：

```
[root@linux-node1 ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:1vTe9j2pgi4A+2Hnj37mrjOG1CwsH1QLVF0aMIvDFyA root@linux-node1.example.com
The key's randomart image is:
+---[RSA 2048]----+
|   E.oo=o...     |
|    o...+.o      |
|     +oo...      |
|   . .o. o .     |
|    = o S . .    |
|   o O =   . .   |
|    * O   . . o .|
|     + *.+ . . +.|
|      o+%=  ... +|
+----[SHA256]-----+
[root@linux-node1 ~]# ssh-copy-id linux-node1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/root/.ssh/id_rsa.pub"
The authenticity of host 'linux-node1 (192.168.56.11)' can't be established.
ECDSA key fingerprint is SHA256:jxXA6iUlGjyXbcXF3HG1zS+wv6qdECYbq0Q2zIJTvoU.
ECDSA key fingerprint is MD5:c3:29:4b:6c:ad:32:c4:dd:1c:c6:97:c0:e0:76:a9:86.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
root@linux-node1's password:
Permission denied, please try again.
root@linux-node1's password:
Permission denied, please try again.
root@linux-node1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'linux-node1'"
and check to make sure that only the key(s) you wanted were added.
```

```
[root@linux-node1 ~]# ssh-copy-id linux-node2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/root/.ssh/id_rsa.pub"
The authenticity of host 'linux-node2 (192.168.56.12)' can't be established.
ECDSA key fingerprint is SHA256:jxXA6iUlGjyXbcXF3HG1zS+wv6qdECYbq0Q2zIJTvoU.
ECDSA key fingerprint is MD5:c3:29:4b:6c:ad:32:c4:dd:1c:c6:97:c0:e0:76:a9:86.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
root@linux-node2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'linux-node2'"
and check to make sure that only the key(s) you wanted were added.
[root@linux-node1 ~]# vim /etc/salt/roster
# 增加如下内容
linux-node1:
  host: 192.168.56.11
  user: root
  priv: /root/.ssh/id_rsa
linux-node2:
  host: 192.168.56.12
  user: root
  priv: /root/.ssh/id_rsa
```

执行测试命令

```
[root@linux-node1 ~]# salt-ssh "*" -r 'df -h'
linux-node2:
    ----------
    retcode:
        0
    stderr:
    stdout:
        文件系统                 容量  已用  可用 已用% 挂载点
        devtmpfs                 899M     0  899M    0% /dev
        tmpfs                    910M   40K  910M    1% /dev/shm
        tmpfs                    910M  9.6M  901M    2% /run
        tmpfs                    910M     0  910M    0% /sys/fs/cgroup
        /dev/mapper/centos-root   39G  2.6G   36G    7% /
        /dev/sda1               1014M  321M  694M   32% /boot
        /dev/mapper/centos-home   19G   33M   19G    1% /home
        tmpfs                    182M     0  182M    0% /run/user/0
linux-node1:
```

```
    ----------
    retcode:
        0
    stderr:
    stdout:
        文件系统                 容量  已用  可用 已用% 挂载点
        devtmpfs                 899M     0  899M    0% /dev
        tmpfs                    910M  264K  910M    1% /dev/shm
        tmpfs                    910M  9.6M  901M    2% /run
        tmpfs                    910M     0  910M    0% /sys/fs/cgroup
        /dev/mapper/centos-root   39G  2.6G   36G    7% /
        /dev/sda1               1014M  321M  694M   32% /boot
        /dev/mapper/centos-home   19G   33M   19G    1% /home
        tmpfs                    182M     0  182M    0% /run/user/0
```

命令格式说明：`salt-ssh "*" -r 'df -h'`

`salt-ssh` 为执行命令， `*` 代表执行目标，`*`指所有节点，由于 `*` 在命令里面有特殊的含义，所以需要增加双引号或者单引号， `-r` 直接执行 shell 命令，不使用模块方式，`'df -h'` 所执行的命令

salt-ssh 默认为 25 个并发执行。

```
[root@linux-node1 ~]# salt-ssh '*' test.ping
linux-node1:
    True
linux-node2:
    True
[root@linux-node1 ~]# salt-ssh 'linux-node2*' state.sls web.apache
```

Grains

`Grains` 数据是 `minion` 每次启动时采集的静态数据，例如：主机名、IP、硬件类型等(并且支持自定义采集信息的类型)，每次重启 `minion` 均会重新采集一次。

```
[root@linux-node1 ~]# salt 'linux-node2*' grains.items
linux-node2.example.com:
    ----------
    biosreleasedate:
        07/22/2020
    biosversion:
        6.00
    cpu_flags:
        - fpu
        - vme
        - de
```

```
        - pse
        - tsc
        - msr
        - pae
        - mce
        - cx8
        - apic
        - sep
        - mtrr
        - pge
        - mca
        - cmov
        - pat
        - pse36
        - clflush
        - mmx
        - fxsr
        - sse
        - sse2
        - ss
        - syscall
        - nx
        - pdpe1gb
        - rdtscp
        - lm
        - constant_tsc
        - arch_perfmon
        - nopl
        - xtopology
        - tsc_reliable
        - nonstop_tsc
        - eagerfpu
        - pni
        - pclmulqdq
        - ssse3
        - fma
        - cx16
        - pcid
        - sse4_1
        - sse4_2
        - x2apic
        - movbe
        - popcnt
        - tsc_deadline_timer
        - aes
        - xsave
        - avx
        - f16c
        - rdrand
```

```
        - hypervisor
        - lahf_lm
        - abm
        - 3dnowprefetch
        - invpcid_single
        - ssbd
        - ibrs
        - ibpb
        - stibp
        - fsgsbase
        - tsc_adjust
        - bmi1
        - avx2
        - smep
        - bmi2
        - invpcid
        - rdseed
        - adx
        - smap
        - clflushopt
        - xsaveopt
        - xsavec
        - arat
        - md_clear
        - spec_ctrl
        - intel_stibp
        - flush_l1d
        - arch_capabilities
    cpu_model:
        Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
    cpuarch:
        x86_64
    cwd:
        /
    disks:
        - sda
        - sr0
    dns:
        ----------
        domain:
        ip4_nameservers:
            - 114.114.114.114
            - 8.8.8.8
        ip6_nameservers:
        nameservers:
            - 114.114.114.114
            - 8.8.8.8
        options:
        search:
```

```
        - example.com
    sortlist:
domain:
    example.com
fqdn:
    linux-node2.example.com
fqdn_ip4:
    - 192.168.56.12
    - 192.168.56.12
    - 192.168.56.12
fqdn_ip6:
    - fe80::20c:29ff:fefc:602f
    - fe80::20c:29ff:fefc:602f
    - fe80::20c:29ff:fefc:602f
fqdns:
    - linux-node2.example.com
gid:
    0
gpus:
    |_
      ----------
      model:
          SVGA II Adapter
      vendor:
          vmware
groupname:
    root
host:
    linux-node2
hwaddr_interfaces:
    ----------
    eth0:
        00:0c:29:fc:60:2f
    lo:
        00:00:00:00:00:00
id:
    linux-node2.example.com
init:
    systemd
ip4_gw:
    192.168.56.1
ip4_interfaces:
    ----------
    eth0:
        - 192.168.56.12
    lo:
        - 127.0.0.1
ip6_gw:
    False
```

```
    ip6_interfaces:
        ----------
        eth0:
            - fe80::20c:29ff:fefc:602f
        lo:
            - ::1
    ip_gw:
        True
    ip_interfaces:
        ----------
        eth0:
            - 192.168.56.12
            - fe80::20c:29ff:fefc:602f
        lo:
            - 127.0.0.1
            - ::1
    ipv4:
        - 127.0.0.1
        - 192.168.56.12
    ipv6:
        - ::1
        - fe80::20c:29ff:fefc:602f
    kernel:
        Linux
    kernelparams:
        |_
          - BOOT_IMAGE
          - /vmlinuz-3.10.0-1160.6.1.el7.x86_64
        |_
          - root
          - /dev/mapper/centos-root
        |_
          - ro
          - None
        |_
          - crashkernel
          - auto
        |_
          - rd.lvm.lv
          - centos/root
        |_
          - rd.lvm.lv
          - centos/swap
        |_
          - rhgb
          - None
        |_
          - quietnet.ifnames
          - 0
```

```
    |_
      - LANG
      - en_US.UTF-8
kernelrelease:
    3.10.0-1160.6.1.el7.x86_64
kernelversion:
    #1 SMP Tue Nov 17 13:59:11 UTC 2020
locale_info:
    ----------
    defaultencoding:
        UTF-8
    defaultlanguage:
        en_US
    detectedencoding:
        UTF-8
    timezone:
        unknown
localhost:
    linux-node2.example.com
lsb_distrib_codename:
    CentOS Linux 7 (Core)
lsb_distrib_id:
    CentOS Linux
lvm:
    ----------
    centos:
        - home
        - root
        - swap
machine_id:
    310caf7bfc9d483eb3122f5d335eb34d
manufacturer:
    VMware, Inc.
master:
    linux-node1.example.com
mdadm:
mem_total:
    1819
nodename:
    linux-node2.example.com
num_cpus:
    2
num_gpus:
    1
os:
    CentOS
os_family:
    RedHat
osarch:
```

```
    x86_64
oscodename:
    CentOS Linux 7 (Core)
osfinger:
    CentOS Linux-7
osfullname:
    CentOS Linux
osmajorrelease:
    7
osrelease:
    7.9.2009
osrelease_info:
    - 7
    - 9
    - 2009
path:
    /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
pid:
    1869
productname:
    VMware Virtual Platform
ps:
    ps -efHww
pythonexecutable:
    /usr/bin/python3
pythonpath:
    - /usr/bin
    - /usr/lib64/python36.zip
    - /usr/lib64/python3.6
    - /usr/lib64/python3.6/lib-dynload
    - /usr/lib64/python3.6/site-packages
    - /usr/lib/python3.6/site-packages
pythonversion:
    - 3
    - 6
    - 8
    - final
    - 0
saltpath:
    /usr/lib/python3.6/site-packages/salt
saltversion:
    3002.2
saltversioninfo:
    - 3002
    - 2
selinux:
    ----------
    enabled:
        True
```

```
        enforced:
            Enforcing
    serialnumber:
        VMware-56 4d 0d db 8c 25 6f 40-2b 53 24 44 e9 fc 60 2f
    server_id:
        2113442677
    shell:
        /bin/sh
    ssds:
    swap_total:
        2047
    systemd:
        ----------
        features:
            +PAM +AUDIT +SELINUX +IMA -APPARMOR +SMACK +SYSVINIT +UTMP
+LIBCRYPTSETUP +GCRYPT +GNUTLS +ACL +XZ +LZ4 -SECCOMP +BLKID +ELFUTILS +KMOD
+IDN
        version:
            219
    systempath:
        - /usr/local/sbin
        - /usr/local/bin
        - /usr/sbin
        - /usr/bin
    uid:
        0
    username:
        root
    uuid:
        db0d4d56-258c-406f-2b53-2444e9fc602f
    virtual:
        VMware
    zfs_feature_flags:
        False
    zfs_support:
        False
    zmqversion:
        4.1.4
```

自定义 `Grains` 文件

```
[root@linux-node1 ~]# vim /etc/salt/grains
test: hehe
[root@linux-node1 ~]# salt '*' saltutil.sync_grains
linux-node1.example.com:
linux-node2.example.com:
[root@linux-node1 ~]# salt '*' grains.get test
linux-node2.example.com:
linux-node1.example.com:
    hehe
```

Pillar

Pillar是在salt 0.9.8 版本后才添加的功能组件。它跟 grains 的结构一样，也是一个字典格式，数据通过key/value的格式进行存储。在Salt的设计中，Pillar使用独立的加密 sessiion ,所以Pillar可以用来传递敏感的数据，例如ssh-key，加密证书等。

存储位置： 存储在master端存放需要提供给 minion 的信息

应用场景：

敏感信息：每个 minion 只能访问 master 分配给自己的信息

自定义 `Pillar` 文件

```
[root@linux-node1 ~]# mkdir /srv/pillar
[root@linux-node1 ~]# cd /srv/pillar/
[root@linux-node1 pillar]# vim apache.sls
{% if grains['os'] == 'CentOS' %}
  apache: httpd
{% elif grains['os'] == 'Debian' %}
  apache: apache2
{% endif %}
[root@linux-node1 pillar]# vim top.sls
# 设置 minion 节点访问
base:
  'linux-node2.example.com':
    - apache
```

注：当前 apache.sls 为 pillar 所使用文件，与 /srv/salt/base/web/ 创建的 apache.sls 并非一个文件。

执行 `Pillar` 文件

```
[root@linux-node1 pillar]# salt '*' pillar.items
linux-node1.example.com:
    ----------
linux-node2.example.com:
    ----------
    apache:
        httpd
```

By: PingQQ