**自动化运维saltstack**

**一、基础介绍**  
==========================================================================================  
**saltstack简单介绍**

**Saltstack是一种全新的基础设施管理方式，是一个服务器基础架构集中化管理平台，几分钟内便可运行起来，速度够快，服务器之间秒级通讯，扩展性好，很容易批量管理上万台服务器，显著降低人力与运维成本；它具备配置管理、远程执行、监控等功能，一般可以理解为简化版的puppet和加强版的func；通过部署SaltStack环境，可以在成千上万台服务器上做到批量执行命令，根据不同业务特性进行配置集中化管理、分发文件、采集服务器数据、操作系统基础及软件包管理等，SaltStack是运维人员提高工作效率、规范业务配置与操作的利器。SaltStack基于Python语言实现，结合轻量级消息队列（ZeroMQ）（SaltStack的通信模式总共分为2种模式：ZeroMQ、REAT，鉴于REAT目前还不是太稳定，通常会选择ZeroMQ模式）与Python第三方模块（Pyzmq、PyCrypto、Pyjinjia2、python-msgpack和PyYAML等）构建。**

**Saltstack运行模式**

**Local：本地,一台机器玩，不建议**

**Master/Minion：通过server/agent的方式进行管理，效率很高（批量管理1000台机器，25秒搞定）**

**Salt SSH：通过SSH方式进行管理，效率相对来说比较低（批量管理1000台机器，83秒搞定）**

**Saltstack三大功能**

**远程执行（执行远程命令）**

**配置管理（状态管理）**

**云管理**

**Saltstack特征**

**1）部署简单、方便；**

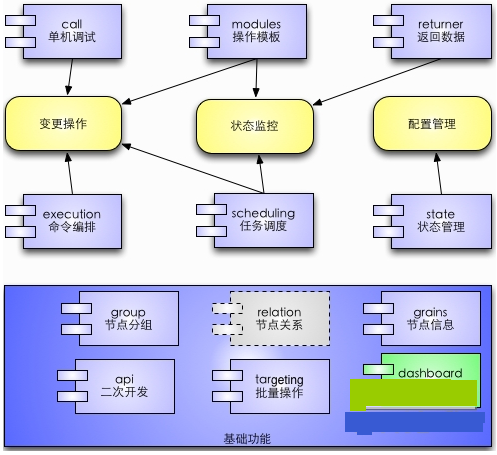
**2）支持大部分UNIX/Linux及Windows环境；**

**3）主从集中化管理；**

**4）配置简单、功能强大、扩展性强；**

**5）主控端（master）和被控端（minion）基于证书认证，安全可靠；**

**6）支持API及自定义模块，可通过Python轻松扩展。**

****

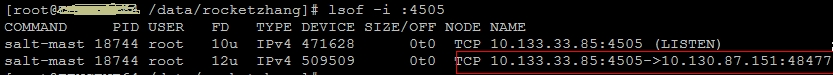
**Master与Minion认证**

**1）minion在第一次启动时，会在/etc/salt/pki/minion/（该路径在/etc/salt/minion里面设置）下自动生成minion.pem（private key）和 minion.pub（public key），然后将 minion.pub发送给master。**

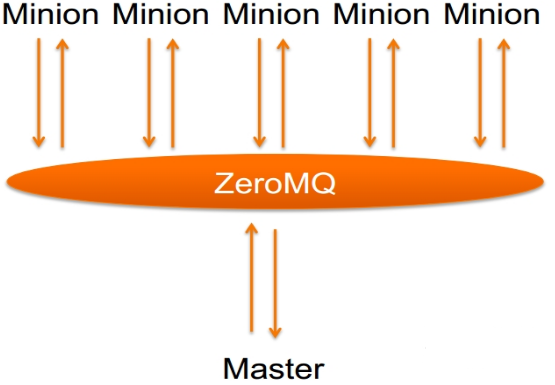
**2）master在接收到minion的public key后，通过salt-key命令accept minion public key，这样在master的/etc/salt/pki/master/minions下的将会存放以minion id命名的 public key，然后master就能对minion发送指令了。**

**Master与Minion的连接**

**1）SaltStack master启动后默认监听4505和4506两个端口。4505（publish\_port）为saltstack的消息发布系统，4506（ret\_port）为saltstack客户端与服务端通信的端口。如果使用lsof 查看4505端口，会发现所有的minion在4505端口持续保持在ESTABLISHED状态。**

****

**2）minion与master之间的通信模式如下**



**二、SaltStack 安装配置过程**

**服务端：xuegod63:**

**IP:192.168.0.63**

**Hostname:xuegod63.com**

**被控端：xuegod64**

**IP:192.168.0.64**

**Hostname:xuegod64.com**

**1、配置FQDN，让ip和主机名对应**

**[root@xuegod63 ~]# vim /etc/hosts**

**192.168.0.63 xuegod63 xuegod63.com**

**192.168.0.64 xuegod64 xuegod64.com**

**[root@xuegod64 ~]# cat /etc/hosts**

**192.168.0.64 xuegod64 xuegod64.com**

**192.168.0.63 xuegod63 xuegod63.com**

**2、两台机器都关闭防火墙和selinux**

**3、服务组件安装：**

**[root@xuegod63 ~]# yum install epel-release -y #安装 epel源**

**[root@xuegod63 ~]# yum install salt-master salt-minion -y #安装服务端和被监控端服务**

**[root@xuegod64 ~]# yum install epel-release -y && yum install salt-minion –y**

**#被控端：安装被监控端服务**

**4、服务启动：**

**配置主控服务：**

**[root@xuegod63 ~]# vim /etc/salt/minion**

**在16行下面增加一行：**

**master: 192.168.0.63 ##说明服务端的地址**

**修改python编写的配置文件千万注意缩进（空格）**

**启动主控端服务：**

**[root@xuegod63 ~]# systemctl start salt-master**

**[root@xuegod63 ~]# systemctl start salt-minion**

**配置被控端服务：**

**[root@xuegod64 ~]# vim /etc/salt/minion**

**在16行下面增加一行：**

**master: 192.168.0.63 ##指定服务端的地址**

**启动被监控服务：**

**[root@xuegod64 ~]# systemctl start salt-minion**

**启动完，可以查看下进程，可以看到相关服务是python来执行运行的**

**[root@xuegod63 ~]# ps aux|grep salt**

**root 1772 0.0 2.0 288104 21088 ? S 09:46 0:00 /usr/bin/python2.6 /usr/bin/salt-master –d**

**root 2166 0.1 2.5 442228 25216 ? S 09:46 0:00 /usr/bin/python2.6 /usr/bin/salt-minion –d**

**5、配置认证**

**下面认证操作都是在xuegod63上操作：**

**salt-key -a xuegod64.com ##增加一个xuegod64的密钥认证**

**[root@xuegod63 ~]# salt-key -a xuegod64.com**

**The following keys are going to be accepted:**

**Unaccepted Keys:**

**xuegod64.com**

**Proceed? [n/Y] y**

**Key for minion xuegod64.com accepted.**

**查看已经签名的客户端：**

**[root@xuegod63 ~]# salt-key -A**

**Accepted Keys:**

**xuegod64.com**

**Denied Keys:**

**Unaccepted Keys:**

**xuegod63.com**

**Rejected Keys:**

**签名完成，可以在被监控端看到生成的密钥minion\_master.pub**

**[root@xuegod64 ~]# cd /etc/salt/pki/minion/**

**[root@xuegod64 minion]# ls**

**minion\_master.pub minion.pem minion.pub**

**补充: salt-key 可以使用-A 来签名所有主机，也可以使用-d 删除指定的主机**

**6、saltstack 远程执行命令演示：**

**探测主机是否在线**

**[root@xuegod63 ~]# salt '\*'test.ping**

**xuegod64.com:**

**True**

**查看分区等信息：**

**[root@xuegod63 ~]# salt'xuegod64.com' cmd.run 'df -h'**

**xuegod64.com:**

**Filesystem Size Used Avail Use% Mounted on**

**/dev/sda2 9.5G 2.0G 7.1G 22% /**

**tmpfs 491M 12K 491M 1% /dev/shm**

**/dev/sda1 2.9G 39M 2.7G 2% /boot**

**/dev/sda3 5.7G 12M 5.4G 1% /usr/local**

**可以调用cmd.run 来执行其他的命令，比如负载，时间，主机名等等**

**这里的\* 表示所有已经签名的客户端，也可以指定其中特定的一个，比如 xuegod64.com，但是这些客户端必须是在master上被接受认证的客户端，**

**如果有多个被监控的客户端，我们可以通过通配符，列表，正则等来一次性的控制多个主机，比如我们现在有两个客户端，web10 web11.可以写成：**

**salt ‘web\*’或者salt ‘web1[01]’或者 salt -L ‘web10，web11’使用列表的时候，多个机器用逗号隔开**

**7、saltstack之grains介绍**

**GRAINS 组件是saltstack中非常重要的一个组件，其主要用于记录Minion的一些静态信息，如比：CPU、内存、磁盘、网络等。grains信息是每次客户端启动后自动上报给master的，一旦这些静态信息发生改变需要重启minion 或者 重新同步下 grains。**

**grains 是在minion启动收集到的一些信息，比如操作系统的类型，网卡IP,内存版本，CPU信息等**

**[root@xuegod63 ~]# salt 'xuegod64.com' grains.ls ##列出所有的grains项目名称**

**xuegod64.com:**

**- SSDs**

**- biosreleasedate**

**- biosversion**

**- cpu\_flags**

**- cpu\_model**

**- cpuarch**

**等等**

**[root@xuegod63 ~]# salt 'xuegod64.com' grains.items ##列出所有的grains项和对应的值**

**xuegod64.com:**

**----------**

**SSDs:**

**biosreleasedate:**

**05/20/2014**

**biosversion:**

**6.00**

**cpu\_flags:**

**- fpu**

**- vme**

**- de**

**启动grains的信息并不是动态的，并不会时时变更，它只是在minion启动时收集到的，我们可以根据grains收集到的一些信息，做配置管理工作，可以支持自定义一些监控项**

**8、自定义grains**

**在被监控端xuegod64上 添加grains：**

**[root@xuegod64 ~]# vim /etc/salt/grains ##此文件默认没有，需要创建**

**添加如下信息，监控项：对应的信息**

**role: zabbix\_server**

**env: test**

**myname: xuegodlinux**

**myhostname: xuegod64.com**

**重启minion服务**

**[root@xuegod64 ~]# systemctl restart salt-minion**

**可以再次执行salt 'xuegod64.com' grains.items 查看是否存在自定义的项，或者使用**

**[root@xuegod63 ~]# salt 'xuegod64.com' grains.item env 单独查看**

**我们可以借助grains的一些属性信息来执行一些命令**

**[root@xuegod63 ~]# salt -G env:test cmd.run 'w'**

**9、saltstack之pillar介绍**

**grains用于存储静态不易变更的数据，而pillar一般用来存储动态的、敏感的数据；通过minion端和master端都可以配置设置或获取grains信息，而pillar信息只能在master端配置，再到minion端执行。**

**pillar和grains不一样，是在master上定义的，并且是针对minion定义的一些信息，可以自定义变量等**

**自定义配置pillar过程：**

**[root@xuegod63 ~]# vim /etc/salt/master**

**找到pillar\_roots 这项，取消下面三行的注释**

**529 pillar\_roots:**

**530 base:**

**531 - /srv/pillar**

**# pillar\_roots 前面加一个空格**

**创建pillar文件的存放目录：**

**[root@xuegod63 ~]# mkdir -p /srv/pillar**

**[root@xuegod63 ~]# cd !$**

**创建配置文件，编辑如下内容：**

**[root@xuegod63 pillar]# vim test.sls**

**conf: /etc/123.conf**

**myname: saltstack-server**

**注意冒号后有空格**

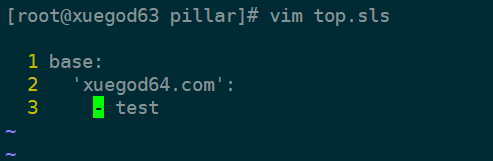
**创建编写入口配置文件**

**[root@xuegod63 ~]# vim /srv/pillar/top.sls**

**base:**

**'xuegod64.com':**

**- test**



**注意- 后有空格**

**重启服务：**

**[root@xuegod63 pillar]# systemctl restart salt-master**

**可以通过刷新来获取新的状态**

**[root@xuegod63 pillar]# salt '\*' saltutil.refresh\_pillar**

**查看：**

**[root@xuegod63 pillar]# salt '\*' pillar.items**

**xuegod64.com:**

**----------**

**conf:**

**/etc/123.conf**

**myname:**

**saltstack-server**

**三、Saltstack 配置管理服务**

1. **配置安装apache**

**[root@xuegod63 ~]# vim /etc/salt/master**

**取消406-408行的注释**

**406 file\_roots:**

**407 base:**

**408 - /srv/salt/**

**创建配置文件存储目录**

**[root@xuegod63 ~]# mkdir /srv/salt**

**[root@xuegod63 ~]#cd /srv/salt**

**创建入口配置文件**

**[root@xuegod63 salt]# vim top.sls**

**base:**

**'xuegod64.com':**

**- apache**

**意思是指定客户端上执行apache的模块**

**重启服务：**

**[root@xuegod63 salt]# systemctl restart salt-master**

**创建编写apache模块的配置文件**

**[root@xuegod63 salt]# vim /srv/salt/apache.sls**

**apache-service:**

**pkg.installed:**

**- names:**

**- httpd**

**- httpd-devel**

**service.running:**

**- name: httpd**

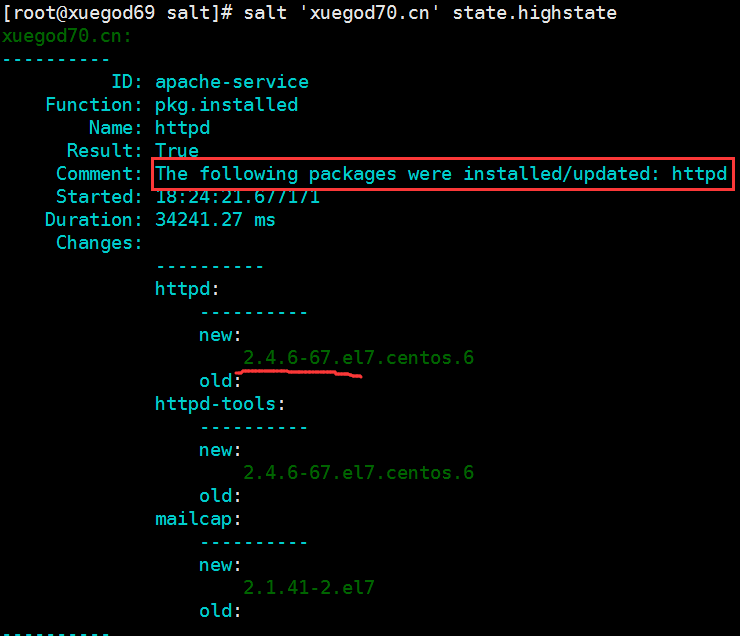
**- enable: True**

***注意：行对齐 -有空格***

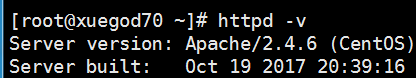
**apache-service是ID的名称，自定义的。pkg.installed为包安装函数，下面是要安装的包的名字。service.running也是一个函数，来保证指定的服务启动，enable表示开机启动。**

**配置好文件，执行**

**[root@xuegod63 salt]# salt 'xuegod64.com' state.highstate**



**查看被控端，已经安装成功。**



**2.配置远程文件管理**

**http://www.comblogs.com/lgeng/p/6567424.html**

**创建一个文件管理的模块**

**[root@xuegod63 salt]# vim filetest.sls**

**file\_test:**

**file.managed:**

**- name: /tmp/xuegodlinux.txt**

**- source: salt://test/xuegod.php**

**- user: root**

**- group: root**

**- mode: 644**

注意：第一行的 file-test 为自定的名字，表示该配置段的名字，可以在别的配置段中引用它；source指定文件从哪里拷贝，这里的 test 目录相当于是 /srv/salt/test 目录；name指定远程客户端要生成的文件。  
  
新建所要测试的源文件

**[root@xuegod63 salt]#mkdir -p /srv/salt/test**

**[root@xuegod63 salt]# echo 'xuegod.com'>/srv/salt/test/xuegod.php**

**在入口文件里添加模块**

**[root@xuegod63 salt]# vim top.sls**

**base:**

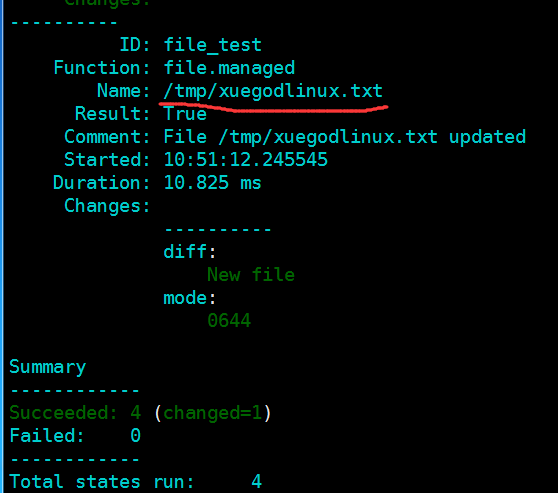
**'xuegod64.com':**

**- apache**

**- filetest**

**测试执行：**

**[root@xuegod63 salt]# salt 'xuegod64.com' state.highstate**



**客户端查看是否有/tmp/xuegodlinux.txt**

**3、配置文件夹管理：**

**创建一个文件夹管理的模块**

**[root@xuegod63 salt]# vim filedir.sls**

**file\_dir:**

**file.recurse:**

**- name: /tmp/testdir**

**- source: salt://test**

**- user: root**

**- file\_mode: 644**

**- dir\_mode: 755**

**- mkdir: True**

**- clean: True**

**在入口文件里添加模块**

**[root@xuegod63 salt]# vim top.sls**

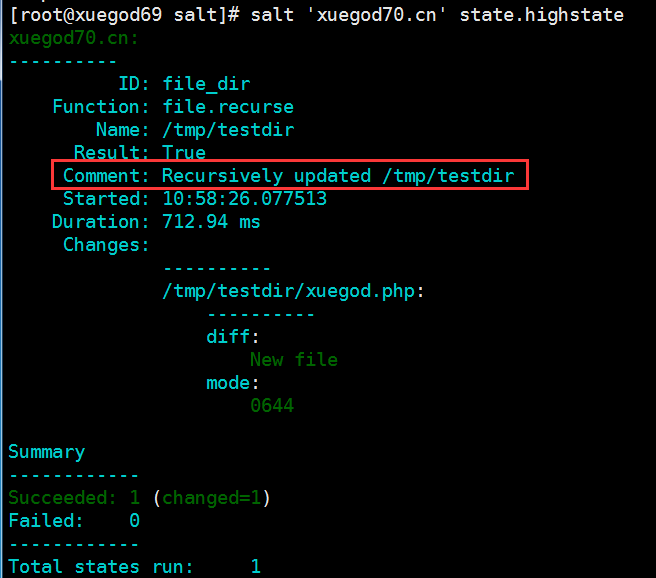
**base:**

**'xuegod64.com':**

**- filedir**

**测试文件夹管理：**

**[root@xuegod63 salt]# salt 'xuegod64.com' state.highstate**



**客户端查看是否有/tmp/testdir 目录**

**4、saltstack远程执行命令**

**创建执行命令的配置文件**

**[root@xuegod63 salt]# vim cmd.sls**

**cmd\_test:**

**cmd.run:**

**- unless: test -f /tmp/xuegodtext.txt**

**- names:**

**- touch /tmp/11.txt**

**- mkdir /tmp/1233**

**- user: root**

注意：unless: 表示若**/tmp/xuegodtext.txt**文件不存在，也就是结果为True则执行-name后面的命令，为false则不执行；还可以使用 onlyif 表示若**/tmp/xuegodtext.txt**文件存在，则执行后面的命令；两者正好相反。

**在入口文件里添加模块**

**[root@xuegod63 salt]# vim top.sls**

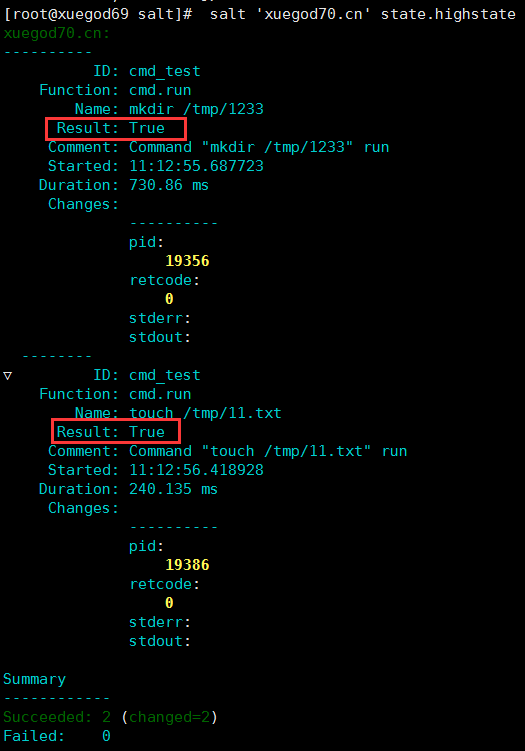
**base:**

**'xuegod64.com':**

**- cmd**

**测试执行命令：**

**[root@xuegod63 salt]# salt 'xuegod64.com' state.highstate**



**5、saltstack远程执行shell脚本**

**[root@xuegod63 salt]# vim shell.sls**

**shell\_test:**

**cmd.script:**

**- source: salt://test/1.sh**

**- user: root**

**[root@xuegod63 salt]# cat test/1.sh**

**#!/bin/bash**

**touch /tmp/111222333.jsp**

**echo 'this is jsp page'>> /tmp/111222333.jsp**

**[root@xuegod63 salt]# vim top.sls**

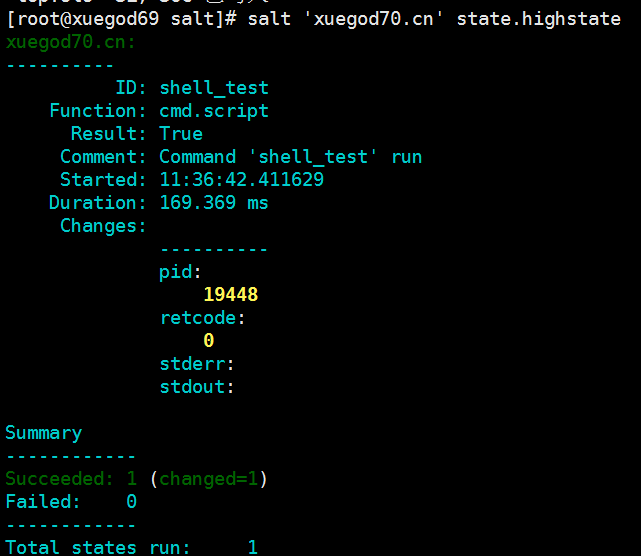
**base:**

**'xuegod64.com':**

**- shell**

**测试执行命令：**

**[root@xuegod63 salt]# salt 'xuegod64.com' state.highstate**



**客户端查看是否执行成功.**

