第 1 章　开始使用 Salt

SaltStack 是由 Thomas Hatch 于 2011 创建的一个开源项目，底层网络架构采用 ZeroMQ 实现。SaltStack 项目的设计初衷是为了实现一个快速的远程执行系统，后来在研发过程中不断加入新的功能，逐渐形成今天强大的 Salt。如今 Salt 的功能已远不止配置管理和远程执行这么简单，Salt 已演变为一个功能强大的平台，在这个平台上，既可以使用 Salt 提供的各种工具来管理服务器基础架构，也可以自己创建工具来满足特定需求。但所有这些强大功能实现的根基仍是 Salt 的远程执行系统（这也正是 SaltStack 的设计哲学）。这一切还是源于当时的设计初衷，所以我们的学习也应该从远程执行开始。

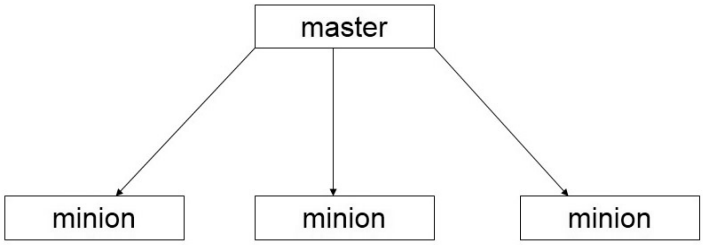
本章学习内容包括：

- Salt 部署的基本架构；
- 安装 Salt——包括软件包安装方式、脚本方式、源码方式及不同平台的安装；
- 配置 Salt——配置 master 服务器和 minion 服务器，建立连接关系；
- 执行第一条远程执行命令。

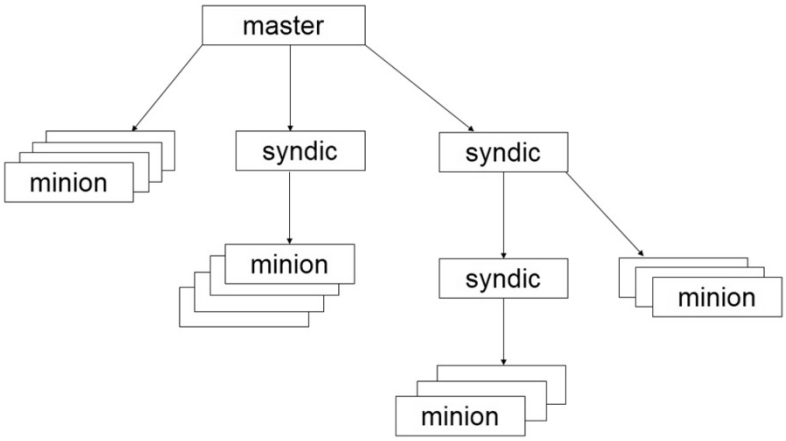
1.1　Salt 部署的基本架构

安装 Salt 前，首先要理解 Salt 基本架构中的各个角色。Salt 架构中最主要的角色是 Salt master 和 Salt minion（另外一种角色是 syndic，将在后面章节详细介绍）。顾名思义，master 是中心控制系统，而 minion 是被管理的客户端。Salt 部署架构可以分成三种。

第一种：master→minion，这种架构中 master 和所有 minion 都直接连接，minion 接收来自 master 的指令，完成命令执行或配置管理，如下图所示。



第二种：master→syndic→minion，这种架构中 master 通过 syndic 对 minion 进行管理，同时该架构可以进行多级扩展，如下图所示。



第三种：无 master 的 minion，这种架构中 minion 不受任何 master 控制，通过本地运行即可完成相关功能。

Salt 的两个主要设计理念是远程执行和配置管理。在远程执行系统中，Salt 用 Python 通过函数调用来完成任务。Salt 中的配置管理系统可以称作 state，也是基于远程执行系统之上，通过 master 的定义可以让对应的 minion 达到想要的系统状态。

理解上面这些基础知识之后，下面我们来学习如何安装 Salt。

1.2　安装 Salt

运行 Salt 所需的环境条件如下所示。

```
python >=2.6 <3.0
zeromq >=2.1.9
pyzmp >=2.1.9
pycrypto
msgpack-python
yaml
jinja2
```

通常解决软件包依赖的最简单方法就是用系统自带的安装包管理器，在 CentOS 系统上为 yum，Ubuntu 系统则为 apt；另外也可以使用 Salt Bootstrap 脚本来安装，Salt Bootstrap 是一个 shell 脚本，该脚本可在各种平台下自动正确完成 Salt 的安装，可自动判断系统类型并采用对应的软件包管理器完成 Salt 的安装和配置。还有一种安装方式就是源码安装，Salt 本身是 Python 程序，可以按照 Python 程序包的标准安装方法进行安装，下面将分别介绍这几种安装方式。

1.2.1　软件包安装方式

操作系统：CentOS 6.5 64 位。

首先安装 EPEL 的 yum 源，默认的仓库是不包含 SaltStack 的，执行如下命令：

rpm-ivh：http://mirrors.zju.edu.cn/epel/6/x86_64/epel-release-6-8.noarch.rpm

安装 epel-release 之后，执行如下命令，可以看到仓库中 SaltStack 相关的包：

```
# yum list|grep salt
python-salttesting.noarch          2015.7.10-1.el6          epel
salt.noarch                        2015.5.5-1.el6           epel
salt-api.noarch                    2015.5.5-1.el6           epel
salt-cloud.noarch                  2015.5.5-1.el6           epel
salt-master.noarch                 2015.5.5-1.el6           epel
salt-minion.noarch                 2015.5.5-1.el6           epel
```

```
salt-ssh.noarch                          2015.5.5-1.el6              epel
salt-syndic.noarch                       2015.5.5-1.el6              epel
```

安装完 EPEL 源之后就可以开始安装 Salt 了，CentOS 系统下安装 SaltStack 非常简单，借助 yum 命令系统可自动解决软件包的依赖问题。

安装 Salt master 时，如果不是 root 用户，命令前请加 sudo，命令如下：

```
# yum -y install salt-master
```

安装 Salt minion 时，如果不是 root 用户，命令前请加 sudo，命令如下：

```
#yum -y install salt-minion
```

安装完毕后可以执行如下命令来查看 Salt 安装信息：

```
salt --versions-report
           Salt: 2015.5.5
         Python: 2.6.6 (r266:84292, Jul 10 2013, 22:43:23)
         Jinja2: 2.2.1
        M2Crypto: 0.20.2
  msgpack-python: 0.4.4
    msgpack-pure: Not Installed
       pycrypto: 2.0.1
        libnacl: Not Installed
         PyYAML: 3.10
          ioflo: Not Installed
          PyZMQ: 14.3.1
           RAET: Not Installed
            ZMQ: 3.2.4
           Mako: Not Installed
        Tornado: 2.2.1
        timelib: Not Installed
       dateutil: 1.4.1
```

1.2.2 脚本安装方式

Salt 官方推荐用 Salt Bootstrap 进行脚本安装，首先下载 bootstrap-salt.sh 脚本：

```
        # wget -c https://raw.githubusercontent.com/saltstack/salt-bootstrap/stable/bootstrap-salt.sh --no-check-certificate
        # sh bootstrap-salt.sh -h 可以查看帮助
        安装 salt-master 和 salt-minion
        # sh bootstrap-salt.sh -M 自动安装 salt-master 和 salt-minion
        单独安装 salt-master
        # sh bootstrap-salt.sh -M -N
        单独安装 salt-minion
        # sh bootstrap-salt.sh
```

1.2.3　源码方式安装

Salt 本身是一个标准的 Python 程序，所以可以通过 pip 命令进行安装，pip 命令可以自动解决 Python 软件包的依赖问题，命令如下：

```
        # pip install salt
```

另一种方式是下载软件包后用 python setup.py install 命令进行安装，首先需要安装官方文档要求的所有依赖软件包，然后下载 Salt 软件包，执行解压后用 python setup.py install 命令进行安装，过程比较烦琐，而且实际中几乎不会采用源码方式安装，所以不再详细介绍，有兴趣的读者可以自己动手实践下。

1.2.4　其他发行版 Linux 系统安装 Salt

有关其他主要发行版本 Linux 上部署 Salt 的方法，可以查看官方文档，都有详细说明。当然，大多数情况下，直接使用 salt-bootstrap 的脚本来安装，更简单方便。参照上面的脚本安装方式相信读者都可以轻易完成不同发行版本的 Linux 下部署 Salt，下面是 salt-bootstrap 脚本支持的 Linux 发行版本：

- Amazon Linux AMI 2012.09；
- Arch Linux；
- CentOS 5/6；
- Debian 6.x/7.x/8（git installations only）；
- Fedora 17/18；

- FreeBSD 9.1/9.2/10；
- Gentoo Linux；
- Linaro；
- Linux Mint 13/14；
- OpenSUSE 12.x；
- Oracle Linux 5/6；
- RHEL 5/6；
- Scientific Linux 5/6；
- SmartOS；
- SuSE 11 SP1 and 11 SP2；
- Ubuntu 10.x/11.x/12.x/13.x/14.x。

小知识：EPEL 是企业版 Linux 附加软件包的简称，是一个由特别兴趣小组创建、维护并管理的，针对红帽企业版 Linux（RHEL）及其衍生发行版（比如 CentOS、Scientific Linux、Oracle Enterprise Linux）的一个高质量附加软件包项目。

1.3 配置 Salt

完成 Salt 的安装后就可以开始配置 Salt 了，Salt master 启动后默认会监听两个端口：4505（publish_port）为 Salt Master pub 接口，提供远程执行命令发送功能；4506（ret_port）为 Salt Master Ret 接口，支持认证（auth），文件服务，结果收集等功能。要确保客户端能跟服务端的这 2 个端口通信，需要保证防火墙对于这两个端口是放开的。在本书的测试环境中防火墙是完全关闭的，读者可以执行 service iptables stop 来关闭防火墙，或者在 iptables 中对这个两个端口放开。Salt master 的配置文件是/etc/salt/master，minion 的配置文件是/etc/salt/minion，这两个配置文件中囊括了大量可调整的参数，这些参数可以控制 master 和 minion 的各个方面，在第 7 章中将详细介绍这些参数。现在我们需要用最简单的配置来完成 master 和 minion 的连接并执行第一条远程命令。

1.3.1　Salt minion 配置

Salt minion 和 master 通信时最好使用域名进行连接，通常在内网环境可以用轻量级的 DNS 软件 dnsmasq 搭建一个域名解析系统，根据 master 和 minion 的主机名和 IP 地址的对应关系设置 dnsmasq，然后将 master 服务器和 minion 服务器的 DNS 地址都设置为装有 dnsmasq 的服务器 IP 地址，这样就可以用域名进行连接。本书中为简便实验环境，直接修改 master 和 minion 的/etc/hosts 文件，步骤如下。

（1）在 master 和 minion 的 hosts 文件最后一行增加下面内容。

```
192.168.1.10 master
192.168.1.11 minion-one
```

（2）用 vi 打开/etc/salt/minion，我们将对该文件进行修改。

首先找出配置选项 master 的命令行，如下所示。

```
#master: salt
```

去掉#，并将 salt 更改为 master，如下所示。

```
master: master
```

如果在文档中找不到相应的命令行，则在该文件的尾部部添加一行。

然后找到 ID 行：

```
#id:
```

去掉#，并将其设置为 minion-one。

```
id: minion-one（这个名字不一定和主机名一样，但最好设置成一样）
```

再次强调下，如果找不到文件中的相应的行，则在该文件的尾部部添加一行，保存并关闭文件。

注意：如果未手动指定 minion ID，minion 将在启动时尝试智能化判断其 minion ID。对于大多数系统，minion ID 将设置为全域名（FQDN）。

1.3.2　启动 Salt master 和 Salt minion

现在我们需要启动（或重启）Salt master 和 Salt minion。如果不是 root 用户，执行下

面的命令时请加上 sudo 命令：

```
# service salt-minion start
# service salt-master start
```

命令执行成功后在 Salt master 上会出现 master 对应进程：

```
/usr/bin/python /usr/bin/salt-master -d
```

在 minion 上会出现 minion 对应进程：

```
/usr/bin/python /usr/bin/salt-minion -d
```

1.3.3 在 master 上接受 minion 秘钥

距离运行第一条 Salt 远程命令，我们仅有一步之遥。minion 在启动后连接 master 会请求 master 为其签发证书，待证书签发完成后，代表 master 可以信任该 minion，并且 Salt 和 master 之间的通信是经过加密的。Salt 配备了 salt-key 命令来帮助我们管理 minion 秘钥。在 Salt master 上执行：

```
# salt-key -L
Accepted Keys:
Unaccepted Keys:
minion-one
Rejected Keys:
```

请注意这时我们的 minion 即 minion-one 这台主机列已经出现在了 Unaccepted Keys 中。这表示该 minion 已经与 master 联系，并且 master 已经获取了 minion 的公钥，正在等待更多进一步有关是否接受该 minion 的指令。

我们可以查看秘钥的指纹码来确保其与 minion 的秘钥相匹配，在 master 上查询的命令如下所示。

```
# salt-key -f minion-one
Unaccepted Keys:
minion-one:  23:63:12:36:3a:9e:e2:55:5a:5d:11:38:91:8c:e1:41
```

我们可以在 minion 上运行 salt-call 命令来获取 minion 的秘钥，在 minion 上查询如下所示。

```
# salt-call --local key.finger
local: 23:63:12:36:3a:9e:e2:55:5a:5d:11:38:91:8c:e1:41
```

指纹码是相互匹配的，所以我们可以在 master 接受该秘钥，如下所示。

```
# salt-key -a minion-one
The following keys are going to be accepted:
Unaccepted Keys:
minion-one
Proceed? [n/Y] Y
Key for minion minion-key accepted.
```

我们可以检验该 minion 秘钥是否已被接受，如下所示。

```
# sudo salt-key -L
Accepted Keys:
minion-one
Unaccepted Keys:
Rejected Keys:
```

对于有很多 minion 的情况，可以在/etc/salt/master 配置文件中查找如下行：

```
#auto_accept: True
```

去掉#，然后保存重启 salt-master，让 master 完成自动签发，另外也可以用 salt-key -A 命令统一接受，这部分在后面章节讲解配置文件和 Salt 常用命令时会有详细说明。

现在准备工作都做好了，可以运行我们的第一条 Salt 命令了！

1.4　第一条命令测试

下面是我们的第一条命令：

```
# sudo salt '*' test.ping
minion-one:
True
```

看上去十分简单，这是一个简单的探测主机是否存活的命令。不用着急，我们很快就会学习更多更有意义的命令。首先来看这条命令，刚才我们执行的这条命令叫远程执行命令。

流程是，我们发送一条消息给一个或所有的 minion，并告诉它们运行 Salt 内置的一个模块中的一条命令（也可以说是模块中的一个函数）。该示例中，minion 返回 true。这个命令能很好地查询我们有哪些 minion 是存活的。下一章中我们会详细探讨 Salt 命令的各个组成部分。

test 模块实际上还包含许多其他有用的函数。怎么样找到这些函数，我们需要使用另外一个模块，如下所示。

```
# salt 'minion-one' sys.list_functions test
minion-one:
- test.arg
- test.arg_repr
- test.arg_type
- test.collatz
- test.conf_test
- test.cross_test
- test.echo
- test.exception
- test.fib
- test.get_opts
- test.kwarg
- test.not_loaded
- test.opts_pkg
- test.outputter
- test.ping
- test.provider
- test.providers
- test.rand_sleep
- test.rand_str
- test.retcode
- test.sleep
- test.stack
- test.tty
- test.version
- test.versions_information
- test.versions_report
```

我们可以来测试一下列表中的其他函数，比如 test.echo，想了解函数的用法可以执行

```
salt minion-one sys.doc test.echo
```

执行结果会给出这个模块的用处和示例用法

```
test.echo:

    Return a string - used for testing the connection

    CLI Example:

        salt '*' test.echo 'foo bar baz quo qux'
```

我们可以根据帮助给出的例子自己写一条命令：

```
salt minion-one test.echo "hello ,I am minion-one"
minion-one:
    hello ,I am minion-one
```

最后用一个有趣的函数结束本章：

```
 salt minion-one sys.doc test.fib
test.fib:

    Return a Fibonacci sequence up to the passed number, and the
    timeit took to compute in seconds. Used for performance tests

    CLI Example:

        salt '*' test.fib 3
```

这个函数是斐波那契数列生成函数，可以生成指定数值范围内的斐波那契数列，斐波那契数列的定义是第 0 项是 0，第 1 项是 1，数列从第三项开始，每一项都等于前两项之和。下面就让我们生成下 50 以内的斐波那契数列吧，命令如下：

```
salt minion-one test.fib 50
minion-one:
    |_
      - 0
```

```
    - 1
    - 1
    - 2
    - 3
    - 5
    - 8
    - 13
    - 21
    - 34
```

本章小结

本章涉及了很多内容，包括在机器上安装了 Salt minion 和 Salt master，并进行最简单的配置完成 master 和 minion 的连接，然后在 master 上接受 minion 的秘钥。我们还运行了第一条 Salt 命令，尽管如此，这些还都只是开始，下一章我们将进一步学习远程执行命令并展示 Salt 到底有多强大。