老男孩教育
oldboyedu.com

云计算与自动化运维实践

自动化运维之SaltStack实践

关于我

- 赵班长
- OpenStack、Zabbix、SaltStack爱好者
- 中国SaltStack用户组(CSSUG) 发起人
- CSSUG：http://www.saltstack.cn
- Blog：http://www.unixhot.com
- Github: http://github.com/unixhot

IT管理的三大核心要素

- People
  - 人员/组织架构
- Process
  - 流程
- Tech/Tool
  - 技术/工具

自动化？

标准化？

工具化？

服务化？

可视化？

版权声明和感谢

◆本课程使用的一些图片以及相关资料均来自于SaltStack官方和互联网，版权归原作者所有。

◆感谢中国SaltStack用户组对SaltStack的大力支持和贡献。

课程大纲

- SaltStack 快速入门
- SaltStack 数据系统
- SaltStack 远程执行
- SaltStack 配置管理
- SaltStack 管理实践
- SaltStack 实践案例
- SaltStack 实现OpenStack自动化部署

第三课 SaltStack 快速入门-配置管理

自动化运维之SaltStack实践

1./etc/salt/master
file_roots:

2.top.sls

3.编写sls

◆ SaltStack是一个新的基础平台管理工具，只需要花费数分钟即可运行起来，可以支撑管理上万台服务器的规模，数秒钟即可完成数据传递。

◆ SaltStack是使用Python语言开发的，同时提供Rest API方便二次开发以及和其它平台进行集成，同时官方也发布了一个Web管理界面halite。

SaltStack常用网址：

- 官方网站：http://www.saltstack.com
- 官方文档：http://docs.saltstack.com
- GitHub：https://github.com/saltstack
- 中国SaltStack用户组：http://www.saltstack.cn

SaltStack运行方式

- Local
- Master/Minion
- Salt SSH

SaltStack三大功能

- 远程执行
- 配置管理
- 云管理

SaltStack支持的操作系统

- CentOS
- RedHat
- Debian
- Ubuntu
- FreeBSD
- Solaris
- Fedora
- Gentoo
- Gentoo
- MAC OS X
- Archlinux
- Windows
- Suse

SaltStack部署

For RHEL/CentOS 5

rpm -ivh http://mirrors.ustc.edu.cn/fedora/epel/5/x86_64/epel-release-5-4.noarch.rpm

For RHEL/CentOS 6

rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm

Salt Master安装：[root@master ~]#yum install -y salt-master

Salt Master启动：[root@master ~]#/etc/init.d/salt-master start

Salt Master加入开机启动：[root@master ~]# chkconfig salt-master on

Salt Minion安装：[root@minion ~]#yum install -y salt-minion

Salt Minion 启动：[root@minion ~]#/etc/init.d/salt-minion start

Salt Minion 加入开机启动：[root@minion ~]#chkconfig salt-minion on

- Grains
- Pillar

Pillar

Grains VS Pillar

名称	存储位置	数据类型	数据采集更新方式	应用
Grains	Minion端	静态数据	Minion启动时收集，也可以使用saltutil.sync_grains进行刷新。	存储Minion基本数据。比如用于匹配Minion，自身数据可以用来做资产管理等。
Pillar	Master端	动态数据	在Master端定义，指定给对应的Minion。可以使用saltutil.refresh_pillar刷新	存储Master指定的数据，只有指定的Minion可以看到。用于敏感数据保存

- 目标（Targeting）
- 模块（Module）
- 返回（Returnners）

- SLS（YAML、Jinja）
- Highstate
- States Module

第四课 SaltStack 数据系统-Grains

Minion启动时收集（静态数据）

Grains应用场景

- Grains可以在state系统中使用，用于配置管理模块。
- Grains可以target中使用，在用来匹配Minion，比如匹配操作系统，使用-G选项。
- Grains可以用于信息查询，Grains保存着收集到的客户端的详细信息。

自动化运维之SaltStack实践

- 用于信息查询
- 在Target中匹配Minion

  –G选项

```
keepalived-server:
    file.managed:
        - name: /etc/keepalived/keepalived.conf
        - source: salt://cluster/files/haproxy-outside-keepalived.conf
        - mode: 644
        - user: root
        - group: root
        - template: jinja
        {% if grains['fqdn'] == 'saltstack-node1.example.com' %}
        - ROUTEID: haproxy_ha
        - STATEID: MASTER
        - PRIORITYID: 150
        {% elif grains['fqdn'] == 'saltstack-node2.example.com' %}
        - ROUTEID: haproxy_ha
        - STATEID: BACKUP
        - PRIORITYID: 100
          {% endif %}
```

第五课 SaltStack 数据系统-Pillar

Salt 0.9.8版本增加了pillar（动态数据）

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

第六课 SaltStack 数据系统
Grains VS Pillar

自动化运维之SaltStack实践

名称	存储位置	数据类型	数据采集更新方式	应用
Grains	Minion端	静态数据	Minion启动时收集，也可以使用saltutil.sync_grains进行刷新。	存储Minion基本数据。比如用于匹配Minion，自身数据可以用来做资产管理等。
Pillar	Master端	动态数据	在Master端定义，指定给对应的Minion。可以使用saltutil.refresh_pillar刷新	存储Master指定的数据，只有指定的Minion可以看到。用于敏感数据保存

Grains VS Pillar

第七课 SaltStack 远程执行-Targeting

- 目标（Targeting）
- 模块（Module）
- 返回（Returnners）

redis-node1-redis03-idc04-soa.example.com

- redis-node1：运行的服务是Redis，这个是第一个节点
- redis03:说明这个redis是Redis集群编号03里面的节点。
- idc04：这台服务器运行在编号04的IDC机房中。
- soa:这台服务器是给SOA服务使用的。
- example.com这台服务是example.com业务

自动化运维之SaltStack实践

和Minion ID有关，需要使用Minion ID：

- Globbing （通配符）
- regex（正则表达式）
- List（列表）

和MinionID无关，不涉及到Minion ID：

- 子网/IP地址
- Grains
- Pillar
- Compound matchers（复合匹配）
- Node groups（节点组）
- Batching execution（批处理执行）

自动化运维之SaltStack实践

Letter	含义	例子
G	Grains glob匹配	G@os:Ubuntu
E	PCRE Minion id匹配	E@web\d+\.(dev|qa|prod)\.loc
P	Grains PCRE匹配	P@os:(RedHat|Fedora|CentOS)
L	minions列表	L@minion1.example.com,minion3.domain.com or bl*.domain.com
I	Pillar glob匹配	I@pdata:foobar
S	子网/IP地址匹配	S@192.168.1.0/24 or S@192.168.1.100
R	Range cluster匹配	R@%foo.bar
D	Minion Data匹配	D@key:value

第八课 SaltStack 远程执行-模块

- 307++

- Service
- Network

第九课 SaltStack 远程执行-返回

carbon_return	将返回数据发送到carbon接收器
cassandra_return	将返回数据发送到cassandra
couchdb_return	将返回数据发送到couchdb
local	将返回数据发送到本地的测试returner接口
memcache_return	将返回数据发送到Memcached服务器
mongo_future_return	将返回数据发送到Mongodb服务器
mongo_return	将返回数据发送到Mongodb服务器
mysql	将返回数据发送到MySQL服务器
postgres	将返回数据发送到Postgres服务器
redis_return	将返回数据发送到Redis服务器
sentry_return	将返回数据发送到sentry
smtp_return	将返回数据发送到SMTP服务器
sqlite3_return	将返回数据发送到sqlite
syslog_return	将返回数据发送到系统的syslog

返回

1.配置Master
mysql.host: 'salt'
mysql.user: 'salt'
mysql.pass: 'salt'
mysql.db: 'salt'
mysql.port: 3306

2.建立数据库

3. salt '*' test.ping --return mysql

第十课 SaltStack 配置管理-概述

SaltStack 配置管理

file_roots	env	SLS	state模块	State关系	实践案例	项目实战
• 设置状态文件的位置	• Base环境 • 开发、测试、预生产、生产	• YAML • Jinja • 编写技巧	• file • pkg • service • cmd	• require • require_in • watch • watch_in • unless • onlyif	• LAMP • LNMP • Zabbix • Haproxy+keepalived	• OpenStack自动化部署

实践、实践、实践

第十一课 SaltStack 配置管理
SLS编写技巧

```
/etc/resolv.conf:
  file.managed:
    - source: salt://init/files/resolv.conf
    - user: root
    - group: root
    - mode: 644
```

规则一：缩进

- YAML使用一个固定的缩进风格表示数据层结构关系。Salt需要每个缩进级别由两个空格组成。
- 不要使用tabs。

规则二：冒号

YAML　　my_key: my_value

```
first_level_dict_key:
  second_level_dict_key: value_in_second_level_dict
```

IN Python　　{'my_key': 'my_value'}

```
{
    'first_level_dict_key': {
        'second_level_dict_key': 'value_in_second_level_dict'
    }
}
```

规则三：短横线

- 想要表示列表项，使用一个短横杠加一个空格。多个项使用同样的缩进级别作为同一列表的一部分。

```
my_dictionary:
  - list_value_one
  - list_value_two
  - list_value_three
```

```
{'my_dictionary': ['list_value_one', 'list_value_two', 'list_value_three']}
```

第十二课 SaltStack 配置管理
LAMP自动化部署

- httpd
- php
- mysql
- mysql-server
- php-mysql
- php-pdo
- php-cli
- php-common

- httpd.conf
- php.ini
- my.cnf

- httpd
- mysqld

官方文档：http://docs.saltstack.com/en/latest/ref/states/all/salt.states.pkg.html

状态模块：pkg

功　　能：管理软件包状态

常用方法：

- pkg.installed　#确保软件包已安装，如果没有安装就安装。
- pkg.latest　　#确保软件包是最新版本，如果不是，进行升级。
- pkg.remove　#确保软件包已卸载，如果之前已安装，进行卸载。
- pkg.purge　　# 除remove外，也会删除其配置文件

官方文档：http://docs.saltstack.com/en/latest/ref/states/all/salt.states.file.html

状态模块：file

功　　能：管理文件状态

常用方法：

- file.managed　# 保证文件存在并且为对应的状态。
- file.recurse　# 保证目录存在并且为对应状态。
- file.absent　# 确保文件不存在，如果存在就删除。

官方文档：http://docs.saltstack.com/en/latest/ref/states/all/salt.states.service.html

状态模块：service

功　　能：管理服务状态

常用方法：

- service.running　#确保服务处于运行状态,如果没有运行就启动。
- service.enabled　#确保服务开机自动启动。
- service.disabled　#确保服务开机不自动启动。
- service.dead　# 确保服务当前没有运行，如果运行就停止。

第十四课 SaltStack 配置管理
状态间关系

官方文档：http://docs.saltstack.com/en/latest/ref/states/requisites.html

功能名称：requisites

功　　能：处理状态间关系

常用方法：

- require　　#我依赖某个状态
- require_in　　#我被某个状态依赖
- watch　　#我关注某个状态
- watch_in　　# 我被某个状态关注

第十四课 SaltStack 配置管理
使用Jinja2模板

官方网站：http://jinja.pocoo.org/

1-File状态使用template参数
- template: jinja

2-模板文件里面变量使用{{ 名称 }}
{{ PORT }}

3.变量列表
- defaults:
PORT: 8080

官方网站：http://jinja.pocoo.org/

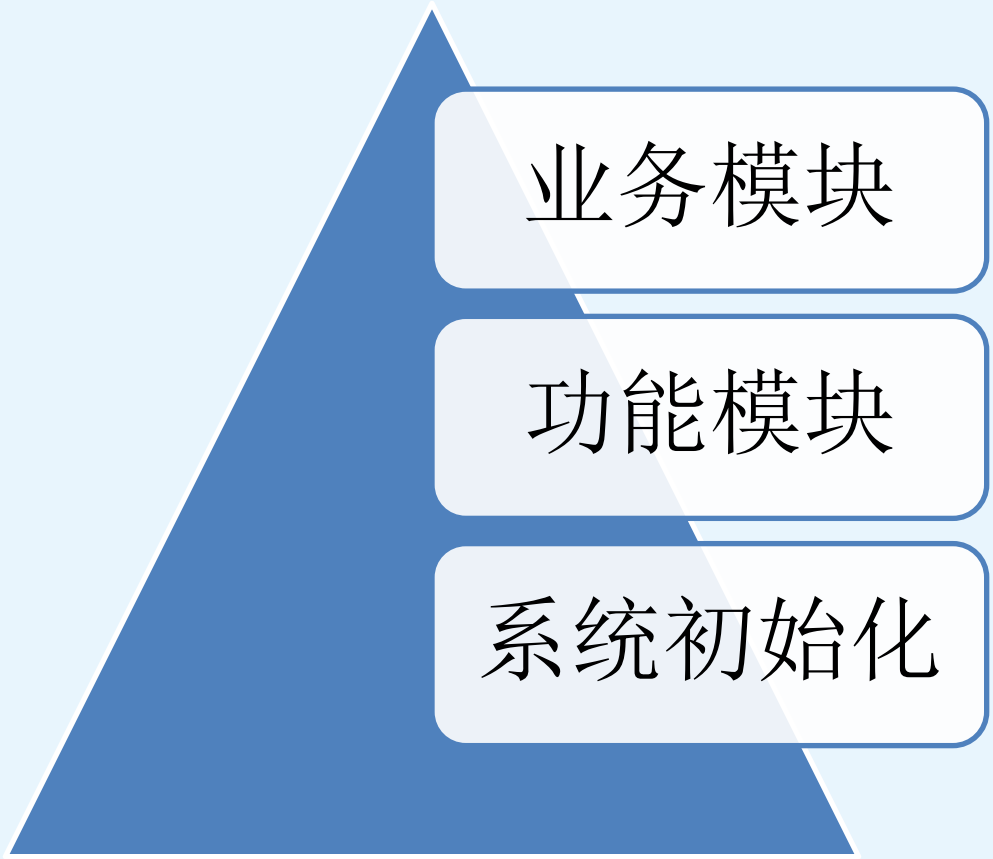
模板文件里面变量使用{{ 名称 }}

- 变量使用Grains： {{ grains[‘fqdn_ip4’] }}
- 变量使用执行模块：{{ salt['network.hw_addr']('eth0') }}
- 变量使用Pillar： {{ pillar[‘apache'][‘PORT'] }}

官方网站：http://jinja.pocoo.org/

```
{% if grains['fqdn'] == 'lb-node1.unixhot.com' %}
  - ROUTEID: HAPROXY_MASTER
  - STATEID: MASTER
  - PRIORITYID: 101
 {% elif grains['fqdn'] == 'lb-node2.unixhot.com' %}
  - ROUTEID: HAPROXY_BACKUP
  - STATEID: BACKUP
  - PRIORITYID: 100
{% endif %}
```

第十五课 SaltStack 实践案例
系统初始化


业务模块
功能模块
系统初始化

历史记录-记录时间

```
/etc/profile:
  file.append:
    - text:
      - export HISTTIMEFORMAT="%F %T `whoami` "
```

记录命令历史到memssages

```
/etc/bashrc:
  file.append:
    - text:
      - export PROMPT_COMMAND='{ msg=$(history 1 | { read x y;
echo $y; });logger "[euid=$(whoami)]":$(who am i):[`pwd`]"$msg"; }'
```

内核参数优化

```
net.ipv4.ip_forward:
  sysctl.present:
    - value: 1
vm.swappiness:
  sysctl.present:
- value: 0
```

```
zabbix-agent:
  pkg.installed:
    - name: zabbix22-agent
  file.managed:
    - name: /etc/zabbix_agentd.conf
    - source: salt://zabbix/files/zabbix_agentd.conf
    - template: jinja
    - defaults:
      Server: {{ pillar['zabbix-agent']['Zabbix_Server'] }}
    - require:
      - pkg: zabbix-agent
  service.running:
    - enable: True
    - watch:
      - pkg: zabbix-agent
      - file: zabbix-agent
```

Zabbix Agent 安装

第十六课 SaltStack 实践案例
Nginx+PHP(FastCGI)配置

官方文档：http://docs.saltstack.com/en/latest/ref/states/all/salt.states.pkg.html

状态模块：cmd

功　　能：执行任意命令

常用方法：

- cmd.run　　#执行任意命令，注意：每次运行状态都会执行。
- cmd.wait　　#当监控的其它状态改变时才执行。
- pkg.remove　#确保软件包已卸载，如果之前已安装，进行卸载。
- pkg.purge　　# 除remove外，也会删除其配置文件

状态模块：状态间关系

功　　能：条件判断，主要用于cmd状态模块

常用方法：

- Onlyif：检查的命令，仅当``onlyif``选项指向的命令返回true时才执行name定义的命令
- unless ：用于检查的命令，仅当``unless``选项指向的命令返回false时才执行name指向的命令


PCRE
• install.sls
Nginx
• install.sls
PHP
• install.sls

```
pcre-source-install:
  file.managed:
    - name: /usr/local/src/pcre-8.37.tar.gz
    - source: salt://nginx/files/pcre-8.37.tar.gz
    - user: root
    - group: root
    - mode: 755
  cmd.run:
    - name: cd /usr/local/src && tar zxf pcre-8.37.tar.gz && cd pcre-8.37 && ./configure --prefix=/usr/local/pcre && make && make install
    - unless: test -d /usr/local/pcre
```

```
nginx-source-install:
  file.managed:
    - name: /usr/local/src/nginx-1.9.1.tar.gz
    - source: salt://nginx/files/nginx-1.9.1.tar.gz
    - user: root
    - group: root
    - mode: 755
  cmd.run:
    - name: cd /usr/local/src && tar zxf nginx-1.9.1.tar.gz && cd nginx-1.9.1 && ./configure --prefix=/usr/local/nginx --user=www --group=www --with-http_ssl_module --with-http_stub_status_module --with-file-aio --with-http_dav_module --with-pcre=/usr/local/src/pcre-8.37 && make && make install
    - require:
      - file: nginx-source-install
    - unless: test -d /usr/local/nginx
```

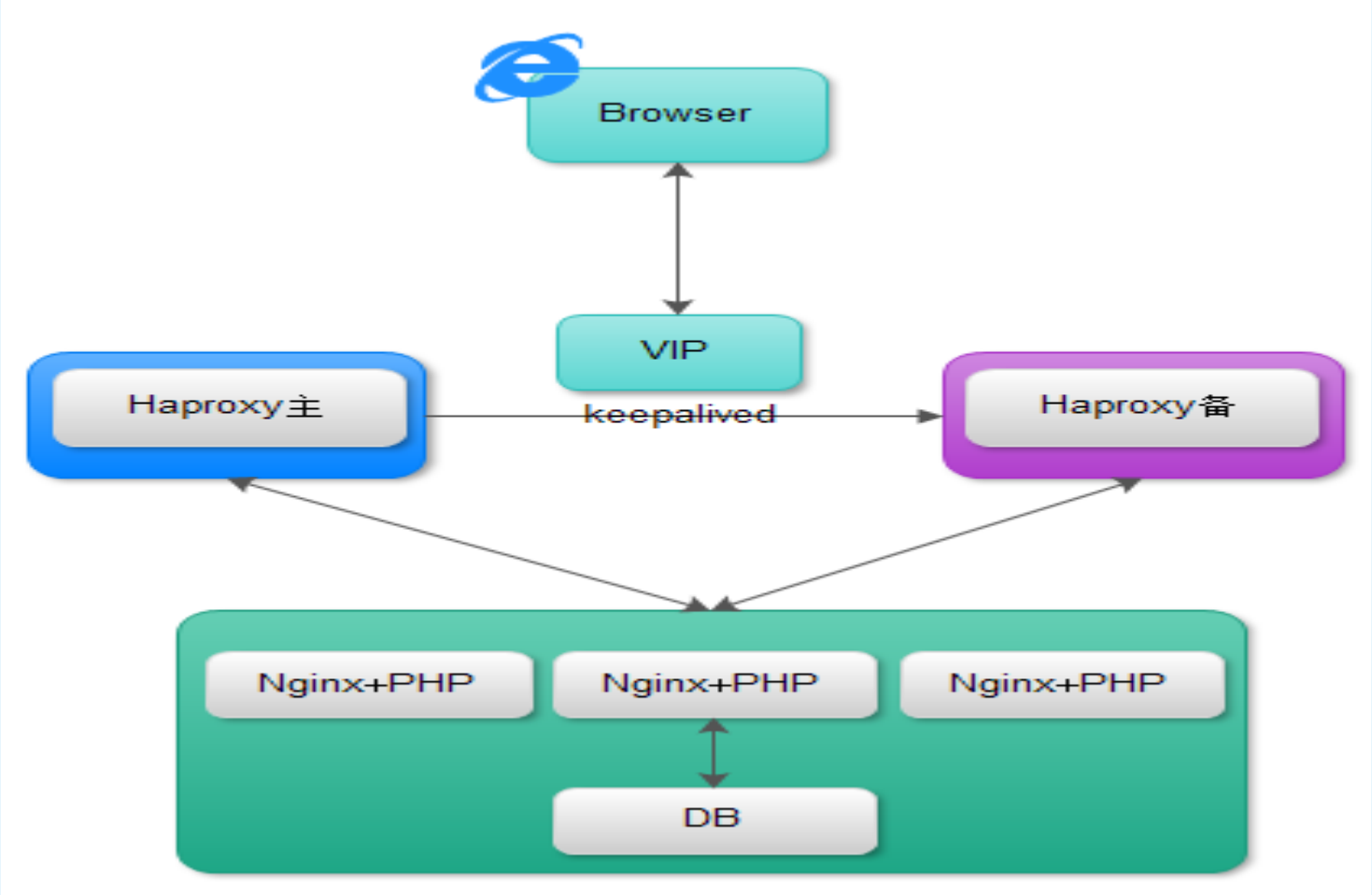
```
php-pkg-init:
  pkg.installed:
    - names:
      - gcc
      - gcc-c++
      - glibc
      - make
      - autoconf
      - libjpeg-turbo
      - libjpeg-turbo-devel
      - libpng
      - libpng-devel
      - freetype
      - freetype-devel
      - libxml2
      - libxml2-devel
      - zlib
      - zlib-devel
      - libcurl
      - libcurl-devel
      - openssl
      - openssl-devel
      - swig
      - mysql
      - mysql-devel
```

```
php-source-install:
  file.managed:
    - name: /usr/local/src/php-5.6.9.tar.gz
    - source: salt://php/files/php-5.6.9.tar.gz
    - user: root
    - group: root
    - mode: 755
  cmd.run:
    - name: cd /usr/local/src && tar zxf php-5.6.9.tar.gz && cd php-5.6.9 &&  ./configure --prefix=/usr/local/php-fastcgi --with-pdo-mysql=mysqlnd --with-mysqli=mysqlnd --with-mysql=mysqlnd --with-jpeg-dir --with-png-dir --with-zlib --enable-xml --with-libxml-dir --with-curl --enable-bcmath --enable-shmop --enable-sysvsem  --enable-inline-optimization --enable-mbregex --with-openssl --enable-mbstring --with-gd --enable-gd-native-ttf --with-freetype-dir=/usr/lib64 --with-gettext=/usr/lib64 --enable-sockets --with-xmlrpc --enable-zip --enable-soap --disable-debug --enable-opcache --enable-zip --with-config-file-path=/usr/local/php-fastcgi/etc --enable-fpm --with-fpm-user=www --with-fpm-group=www && make && make install
    - unless: test -d /usr/local/php-fastcgi
  require:
    - file: php-source-install
    - pkg: php-source-install
```

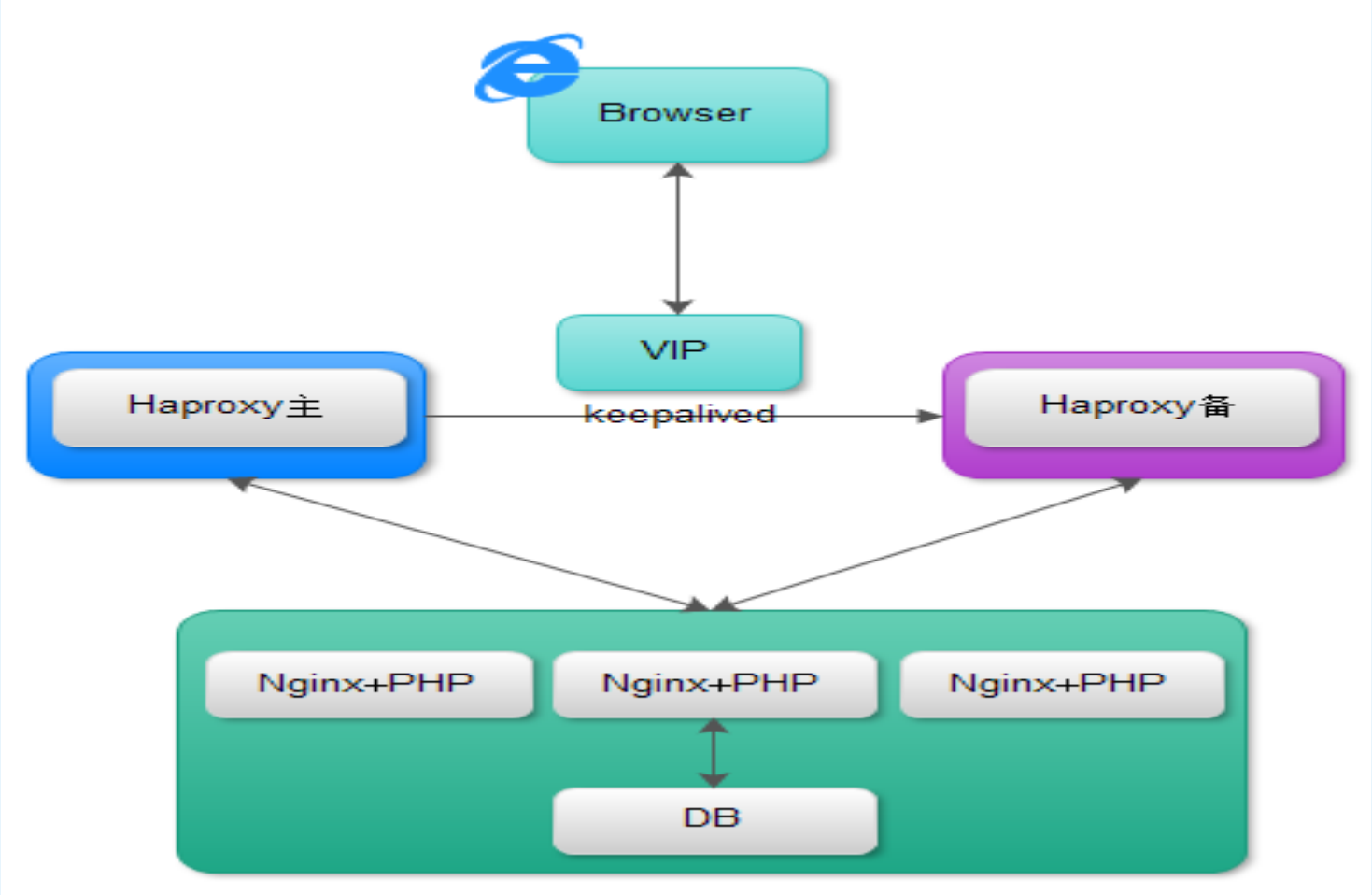
```
pdo-plugin:
  cmd.run:
    - name: cd /usr/local/src/php-5.6.9/ext/pdo_mysql/ && /usr/local/php-fastcgi/bin/phpize && ./configure --with-php-config=/usr/local/php-fastcgi/bin/php-config &&  make && make install
    - unless: test -f /usr/local/php-fastcgi/lib/php/extensions/*/pdo_mysql.so
  require:
- cmd: php-install
```

```
redis-plugin:
  file.managed:
    - name: /usr/local/src/phpredis-2.2.7.tgz
    - source: salt://php/files/phpredis-2.2.7.tgz
    - user: root
    - group: root
    - mode: 755
  cmd.run:
    - name: cd /usr/local/src && tar zxf phpredis-2.2.7.tgz && cd phpredis-2.2.7 && /usr/local/php-fastcgi/bin/phpize && ./configure --with-php-config=/usr/local/php-fastcgi/bin/php-config &&  make && make install
    - unless: test -f /usr/local/php-fastcgi/lib/php/extensions/no-debug-non-zts-20121212/redis.so
  require:
    - file: redis-plugin
- cmd: php-install
```

第十九课 SaltStack 管理实践 Haproxy+Keepalived


Browser
VIP
keepalived
Haproxy主
Haproxy备
Nginx+PHP
Nginx+PHP
Nginx+PHP
DB

第二十课 SaltStack 管理实践 Haproxy+Keepalived（2）


Browser
VIP
keepalived
Haproxy主
Haproxy备
Nginx+PHP
Nginx+PHP
Nginx+PHP
DB

第二十二课 SaltStack 管理实践
Salt SSH

```
<Salt ID>:    # 目标ID
    host:     # 远程主机的IP地址或者主机名
    user:     # 可以登录的用户
    passwd:   # 可以登录用户的密码
    # 可选参数
    port:     # SSH端口
    sudo:     #是否运行sudo,设置True或者False
    priv:     # SSH私钥的路径, 默认是salt-ssh.rsa
    timeout:  # 连接SSH时的超时时间
```

- 运行原始Shell调用：-r
- 状态管理：同salt
- Target：glob及正则

第二十三课 SaltStack 管理实践
Job管理

使用saltutil管理Job

- saltutil.running #查看minion当前正在运行的jobs
- saltutil.find_job<jid> #查看指定jid的job(minion正在运行的jobs)
- saltutil.signal_job<jid> <single> #给指定的jid进程发送信号
- saltutil.term_job <jid> #终止指定的jid进程(信号为15)
- saltutil.kill_job <jid> #终止指定的jid进程(信号为9)

使用Salt Runner管理Job

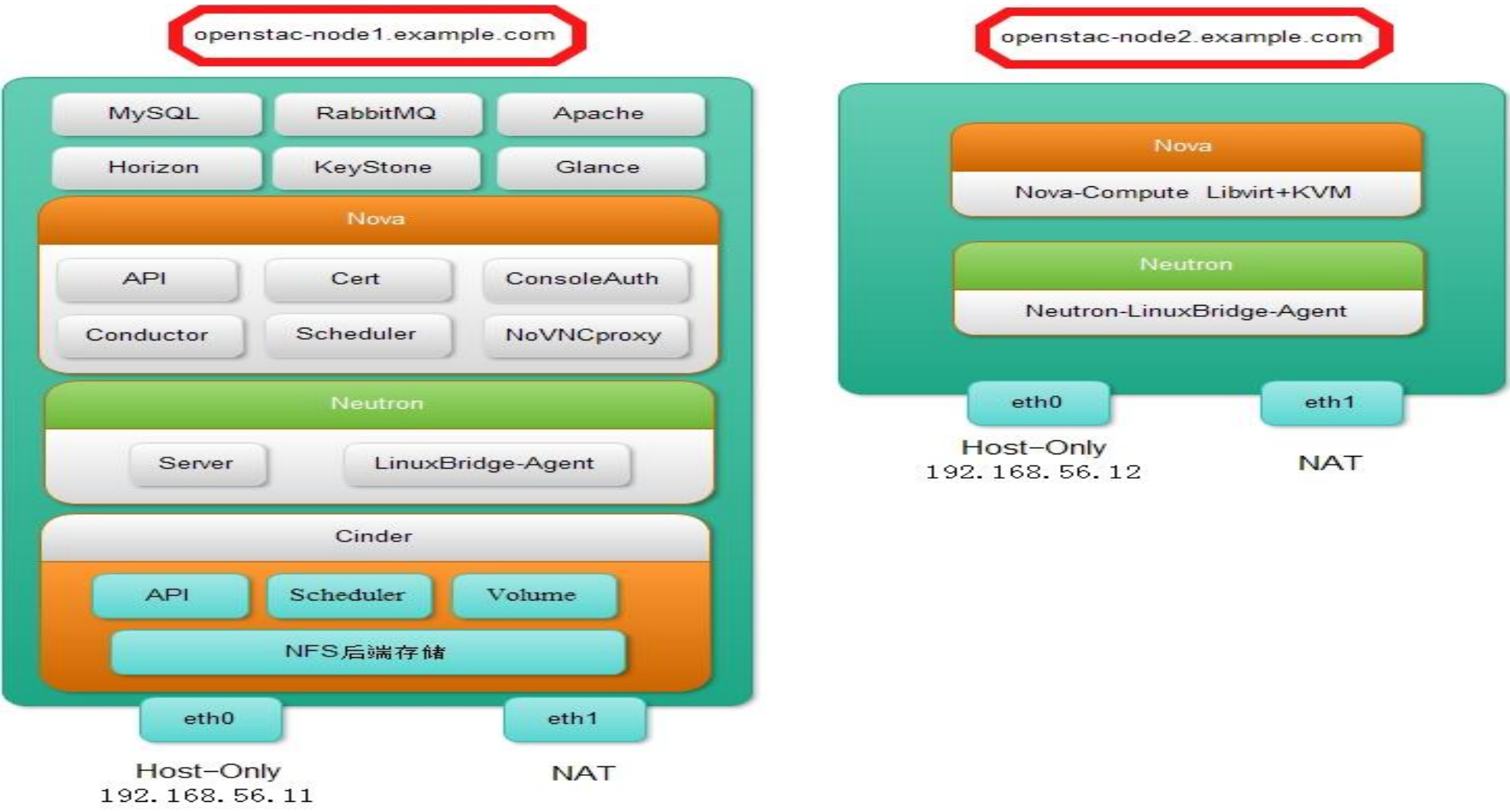
- salt-run jobs.active#查看所有minion当前正在运行的jobs(在所有minions上运行saltutil.running)
- salt-run jobs.lookup_jid<jid> #从master jobs cache中查询指定jid的运行结果
- salt-run jobs.list_jobs#列出当前master jobs cache中的所有job

Salt Schedule

- Minion端（执行模块）
- Master端（Runner）
- Pillar（saltutil.refresh_pillar）

OpenStack自动化部署-架构概述

SaltStack 管理实践



获取demo tenant-id

[root@linux-node1 ~]# keystone tenant-list | awk '/ demo / {print $2}'

创建FLAT网络

[root@linux-node1 ~]#neutron net-create --tenant-id 0034cf7ea3cb4c9aa75a9d922b68f04c flat_net --shared --provider:network_type flat --provider:physical_network physnet1

查看网络

[root@linux-node1 ~]# neutron net-list

创建子网

[root@linux-node1 ~]#neutron subnet-create flat_net --name flat_subnet --allocation-pool start=192.168.56.200,end=192.168.56.250 --enable-dhcp 192.168.56.0/24

查看子网

[root@linux-node1 ~]# neutron subnet-list

获取镜像：wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img

上传镜像：glance image-create --name "cirros-0.3.4-x86_64" --disk-format qcow2 --container-format bare --is-public True --file cirros-0.3.4-x86_64-disk.img

OpenStack自动化部署-源码解析