服务器自动化运维管理方案

目　　录

第 1 章　自动化运维概述 3
1.1. 背景 3
1.2. 自动化运维体系介绍 3
1.3. 开源自动化运维工具简介 4
1.4. 常用自动化运维工具对比选型 5
第 2 章　Puppet 使用说明 7
2.1. Puppet 简介 7
2.2. Puppet 部署说明 8
2.2.1. Puppet 服务器端部署 8
2.2.2. Puppet 客户端安装 8
2.2.3. 证书分发 9
2.3. Puppet 使用说明 9
2.3.1. Puppet Server 架构介绍 9
2.3.2. Puppet 脚本编写规范 9
2.3.3. Puppet 资源 10
2.3.4. 资源使用实例 11
第 3 章　Func 安装使用 15
3.1. Func 介绍 15
3.2. Func 安装与配置 15
3.3. Func 使用说明 17
3.3.1. Func 服务器群组管理 17
3.3.2. Func 模块说明 18
3.4. 常用模块使用说明 18
第 4 章　附录 20
4.1. 参录资料 20
4.2. 招贤纳士 20

第1章 自动化运维概述

1.1. 背景

公司的 XXXX 项目于近期上线，预计会短时间内部署几百组服务器，以运维部目前有限的人手，如果不采取自动化安装、配置及监控方案，工作量可想而知，甚至会出现服务器爆满却来不及上新系统的情况。因此运维部的工程师都有必要熟练掌握自动化运维系统的使用。

1.2. 自动化运维体系介绍

一个完善的自动化运维体系包括系统预备、配置管理以及监控报警 3 个功能模块：

1. **系统预备**
   i. 自动化安装操作系统及常用软件包
2. **配置管理**
   i. 自动化部署业务系统软件包并完成配置
   ii. 远程管理服务器（开关服务等）
   iii. 变更回滚
3. **监控报警**
   i. 服务器可用性、性能、安全监控
   ii. 向管理员发送报警信息

根据提供的功能不同，自动化运维工具也可以分为以下 3 类，如下表所示：

预备类工具	配置管理类	监控报警类
Kickstart	Chef	Nagios
Cobbler	ControlTier	OpenNMS
OpenQRM	Func	Zabbix
Spacewalk	Puppet	Cacti

1. **预备类工具**

预备类工具可以使 Linux 操作系统及软件安装自动化。它们借助服务器上的软件包系统比如 rpm 或者 apt 来安装软件包，甚至会做一些粗略的配置工作。

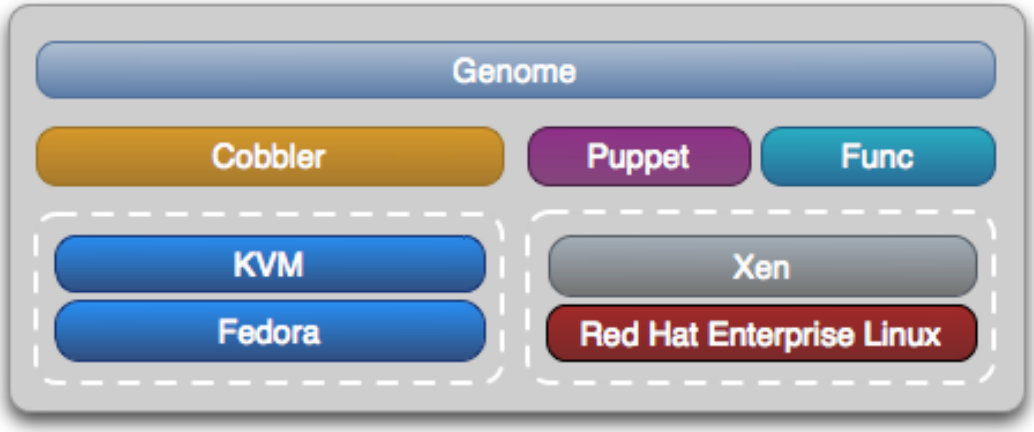
2. 配置管理类工具

配置管理类工具可以自动化部署常用的应用程序，设置参数或者开启一个新服务器上的服务，也可以用来把对操作系统及业务支撑系统的变更管理回滚到上一版本。

3. 监控报警类工具

监控工具用来收集服务器数据，从而生成可用性、性能和其它系统状态的报告。可用性监控可以第一时间向运维人员发送业务不可用的警告，以便第一时间处理，减少业务中断时间。

红帽子资助的 Genome 项目是将预备类、配置管理类集成到一起的框架，如下图所示：



1.3. 开源自动化运维工具简介

预备类工具	配置管理类	监控类
Kickstart	**Chef**	**Nagios**
针对红帽 Linux/Fedora 等发行版的自动化安装方式，简单的讲就是让系统在安装过程中从一个 ks.cfg 配置文件中自动获取所有需要配置的参数。 源于 Anaconda 项目。 技术支持：红帽/Fedora 社区	一个系统集成框架，可以用 Ruby 等代码完成服务器的管理配置并编写自己的库。 技术支持：OpsCode 起始于：2009 年 1 月 官方网站： http://www.opscode.com/chef/	一个强大的监控预警系统，可以监控系统、应用、服务以及各种进程的运行状况，并提供了多种警报机制。 【技术支持】Nagios Enterprises 【发起人】Ethen Galstad 【起始于】1999 年 【官方网站】 http://www.nagios.org/

Cobbler	Cfengine	OpenNMS
为了实现快速网络安装环境的 Linux 安装服务器，可以为数量众多的 Linux 服务自动化执行任务。 【发起人】Michael DeHaan 【技术支持】红帽/Fedora 社区 【起始于】2007 年之前 【官方网站】 https://fedorahosted.org/cobbler/	cfengine 是一个用 C 语言开发的功能强大的自动化系统管理工具.引用其官网的说法"cfengine 是一种 UNIX 管理工具，其目的是使简单的管理的任务自动化，使困难的任务变得较容易。 【起始于】1993 年 【官网网站】http://cfengine.com/	【描述】一个网络管理应用平台，可以自动识别网络服务，事件管理与警报，性能测量等任务。 【技术支持】openNMS group 【起始于】2005 年之前 【官方网站】 http://www.opennms.org/
OpenQRM	**Func**	**Zabbix**
【描述】针对数据中心管理的开源平台，针对设备的部署、监控等多个方面通过可插拔式架构实现自动化的目的，尤其面向云计算/基于虚拟化的业务。 【发起人】Matt Rechenburg 【起始于】2005 年之前 【官方网站】 http://www.openqrm.com/	【描述】全称为 Fedora Unified Network Controller，Fedora 统一网络控制器，用于自化的远程服务器管理。 【发起人】Michael DeHaan 等 【技术支持】红帽/Fedora 社区 【官方网站】 https://fedorahosted.org/func/	【描述】用于监控网络上的服务器/服务以及其他网络设备状态的网络管理系统，后台基于 C，前台由 PHP 编写，可与多种数据库搭配使用。提供各种实时报警机制。 【技术支持】Zabbix 公司 【发起人】Alexei Vladishev 【起始于】1998 年 【官方网站】 http://www.zabbix.com/
Spacewalk	**Puppet**	**Cacti**
【描述】针对红帽/Fedora 等发行版的软件更新管理软件，同时也提供预备和监控的功能。这个项目衍生了红帽 Network Satellite 产品。 【技术支持】红帽 【起始于】2001（Red Hat Network）/2008 【官方网站】 http://spacewalk.redhat.com/	【描述】一个开源的数据中心自动化/配置管理框架，用于 Puppet 自己的声明语言自动化重现任意的系统配置。 【技术支持】Puppet Labs 【官方网站】 http://www.puppetlabs.com/	Cacti 是一套基于 PHP,MySQL,SNMP 及 RRDTool 开发的网络流量监测图形分析工具。它通过 snmpget 来获取数据，使用 RRDtool 绘画图形，它提供了非常强大的数据和用户管理功能，同时也能自己增加模板，功能非常强大完善。 起始于 2004 年 【技术支持】Cacti Group

1.4. 常用自动化运维工具对比选型

预备类工具	
Kickstart	**Cobbler**
供 Anaconda 读取的无人值守安装配置脚本，自动化安装配置过程繁琐	一个集成工具，集成了 PXE、DHCP、DNS 和 Kickstart 服务管理和 yum 仓库管理，简化了运维工程师工作量

<table>
<tr><th colspan="2">集中化配置管理类</th></tr>
<tr><th>Chef</th><th>puppet</th></tr>
<tr><td>1. 需要用户熟悉 ruby 语言，入门门槛高，管理模块开发周期长
2. 资源脚本从前向后执行，维护调试繁琐
3. chef 在配置中心服务器端需要依赖软件比较多，需要 couchdb、RabbitMQ、Solr、java 和 erlang，配置过程复杂繁琐
4. chef 的配置管理文件放在 couchdb 和 solr 索引等二进制文件中，通过远程命令工具 knife 来操作这些配置，维护不方便
5. chef 的用户群少，出了问题不方便排查</td><td>1. Puppet 自有的配置语言较为高级，入门简单，管理模块开发周期短
2. puppet 资源之间有显式的依赖关系，与这些资源在配置文件的位置或前后没有关系
3. puppet 安装简单，需要的支持软件也少，配置过程十分简单
4. puppe 的配置管理文件为 puppet 语言描述的文本文件，易于发布、备份和扩展
5. puppet 的用户很多，Google、Redhat 等大公司都在使用，可以借鉴成熟的经验</td></tr>
<tr><th>Cfengine</th><th>Puppet + Func</th></tr>
<tr><td>1. 老牌的配置管理工具，功能强大，但语法晦涩，学习、维护成本高</td><td>1. 新兴的配置管理工具，语法简单，易于学习、维护，但远程执行命令 Exec 资源只能返回成功与否，执行过程无法跟踪查看，需要和简单易用的 Linux 集群管理工具 Func 配合使用</td></tr>
<tr><th colspan="2">监控类工具</th></tr>
<tr><td colspan="2">监控类工具中有 Zabbix、Nagois 和 cacti 等工具，zabbix 和 Nagois+cacti 组合都是很优秀的工具，鉴于 zabbix 参考资料较少，选择了常用的 Nagois+cacti 组合</td></tr>
</table>

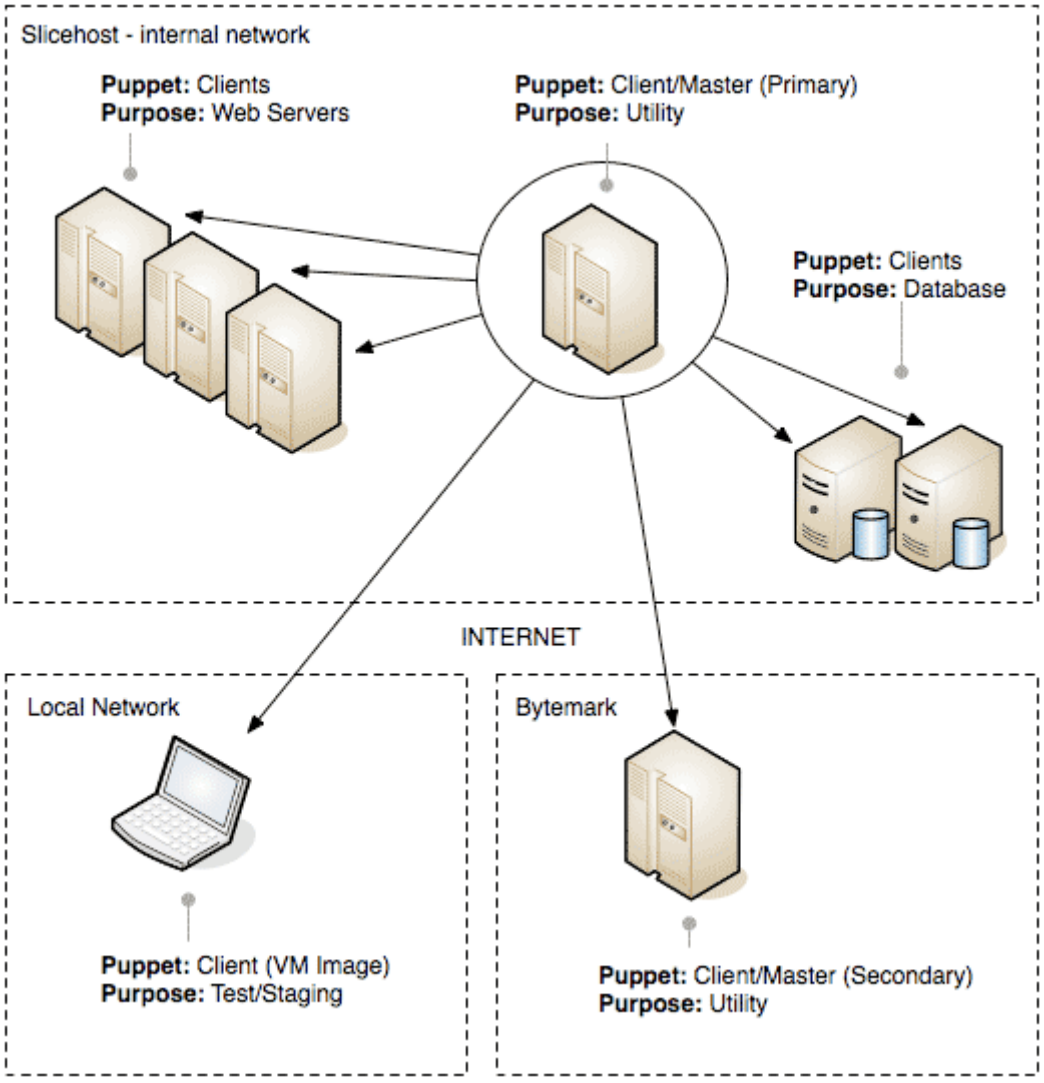
我们的自动化配置管理系统采用了开源的 Puppet 与 Func 的组合，Puppet 用来对服务器及业务系统进行统一配置，Func 用来集中化管理多台服务器。

第2章 Puppet 使用说明

2.1. Puppet 简介

Puppet 是 Puppet Labs 基于 ruby 语言开发的自动化系统配置工具，可以以 C/S 模式或独立模式运行，支持对所有 UNIX 及类 UNIX 系统的配置管理，最新版本也开始支持对 Windows 操作系统有限的一些管理。Puppet 适用于服务器管理的整个过程，比如初始安装、配置、更新以及系统下线。

典型的 Puppet 架构为星型结构，Clients 默认每 30 分钟请求一次 Server 端，确认是否有新的变更操作指令，puppet 支持以节点的方式管理若干的服务器群组，常见的架构图如下：



2.2. Puppet 部署说明

2.2.1.Puppet 服务器端部署

Puppet 服务器端可以安装于常见的 Linux 发行版本中，我们选了陕西西安机房的一台服务器作为了 Puppet 服务器端，操作系统为 Centos 5.4 32 位。

主机名设置为 jvpuppet，并在 DNS 中创建一条指向该 IP 的 A 记录 jvpuppet.jooov.cn

Puppet 的安装方式支持源码安装、yum 和 ruby 的 gem 包安装，Centos 可以直接使用 yum 来安装，Centos 的默认源中没有 puppet，需要先安装 EPEL 包。

Tips：什么是 EPEL

EPEL（http://fedoraproject.org/wiki/EPEL ） 是由 Fedora 社区打造，为 RHEL 及衍生发行版如 CentOS、Scientific Linux 等提供高质量软件包的项目。

安装过程如下：

1. 安装 EPEL：rpm –Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-4.noarch.rpm
2. 安装 puppet 服务端

yum –y install puppet-server

3. 启动 puppet Server

Service puppetmaster start

2.2.2.Puppet 客户端安装

Puppet 服务器端对客户端的管理是基于主机名的，所以在安装之前需要为客户端设置唯一的主机名，主机名命令最好能体现出主机的机房、业务系统名称及 IP 信息。客户端安装的过程如下：

1. 安装 EPEL 包：rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-4.noarch.rpm
2. 安装 puppet 客户端：

```
yum -y install puppet
```

2.2.3.证书分发

Puppet 客户端与服务器端是通过 SSL 隧道通信的，客户端安装完成后，需要向服务器端申请证书：

1. 首次连接服务器端会发起证书申请，命令如下：

   puppetd --server puppet.puppet.cn –test

2. 服务器端可以用 puppetca –list 命令查看到申请证书的客户端主机名
3. puppetca –s 命令可以为特定的主机颁发证书，puppetca –s and –a 表示给所有的主机颁发证书

2.3. Puppet 使用说明

默认安装好的 Puppet 没有任何配置管理功能，需要运维工程师自己开发配置管理功能模块。

2.3.1.Puppet Server 架构介绍

Puppet Server 默认安装完毕后，位于/etc/puppet 目录下，目录结构及简单的描述如下表所示：

```
[jvuser@puppet puppet]$ pwd
/etc/puppet
[jvuser@puppet puppet]$ ls
auth.conf           client 访问 puppet server 的 ACL 配置文件
fileserver.conf     puppet server 作为文件服务器的 ACL 配置文件
manifests           Puppet 脚本主文件目录，至少需要包含 site.pp 文件
modules             Puppet 模块目录，存放 Puppet 脚本的功能模块
namespaceauth.conf  命名空间 ACL 配置文件
puppet.conf         Puppet 服务器端配置文件
```

2.3.2.Puppet 脚本编写规范

Manifests 为 puppet 脚本主目录，site.pp 为程序入口脚本文件，Modules 目录为 puppet 管理脚本模块所在目录，每个模块也有各自的主目录及入口函数文件，如下图所示：

2.3.3.Puppet 资源

Puppet 提供了 48 种资源类型，用户也可以开发一些其他资源，下表列出了常用的几种资源，用这些资源就可以完成大多数系统的配置管理了，如下表所示：

资源类型	描述
文件	文件管理资源，用于管理系统本地文件
组	管理用户组
用户	管理系统用户
包	管理软件包的安装、升级和删除
Yum 库	yum 库配置文件（用 file 资源也可以实现同样的功能）
服务	管理系统服务（启用、禁用和重启等）
Crontab 任务	Crontab 定时任务管理
文件系统挂载	管理已挂载的文件系统（挂载、卸载，维护挂载表）
Zfs	管理 ZFS，在 zfs 实例上创建，删除并并且设置属性
Hosts 主机管理	管理系统主机名（/etc/hosts）
Exec 资源	执行外部命令

2.3.4.资源使用实例

1. File 资源

以下的例子为新建一个/tmp/puppettest 文件，文件内容为：puppet test only，文件权限为 666：

```
class test{
    file { "/tmp/puppettest":
        content => "Puppet test only",
        mode    => 666
    }

}
```

2. User 和 group

Ensure 参数可以创建或者删除组,设置 absent 就删除该组,设置 present 就创建该组，以下的例子为删除不必要的用户和组：

```
$grouplist = [
        "lp","uucp",
        "games","news",
        "floppy","audio"]
    user
    { $userlist:
      ensure => absent,
    }
    group
    { $grouplist:
      ensure => absent,
    }
```

3. Package 资源

为新装的系统安装 ntp 和 screen，删除 pppoe 和 pppoe-conf 包

```
package {
    ["ntp","screen"]:
    ensure => installed;
    ["pppoe","pppoe-conf"]:
    ensure => absent;
    }
```

4. Service 资源

以下实例为启动 ssh 服务，停止 nfs 服务

```
service {
            "ssh":
              ensure => running;
            "nfs":
              ensure => stopped;
          }
```

5. crontab 资源

以下实例摘自公司服务器初始化的一个片断，主要功能为在 crontab 添加时间自动同步任务，同时使用了 package、service 和 crontab 三种资源：

```
class jooov::cron
{
    package
    { "crontabs":
       ensure => installed,
    }
    service
    {
    "crond":
    ensure => running,
    enable => true,
    require => Package["crontabs"];
```

```
    }
        cron
        { ntpdate:
          command => "/usr/sbin/ntpdate time.windows.com",
            user => root,
            hour => 0,
            minute => 0,
      require => Package["crontabs"];
        }
}
```

6. exec 资源

以下代码片断摘自公司服务器初始化配置脚本，主要作用是自动配置初始化 iptables 安全策略：

```
    file
    {
        "/etc/rc.d/foruser.sh":
        ensure => present,
        mode   => 755,
        owner  => root,
        group  => root,
        content => template("iptables/firewall.erb");
    }
    file
    {
        "/etc/rc.d/rc.local":
        ensure => present,
        mode   => 754,
        owner  => root,
        group  => root,
```

```
    content => "touch /var/lock/subsys/local\n/etc/rc.d/firewall.sh\n";
}
 exec {
                "/etc/rc.d/jv_firewall.sh":
    cwd => "/etc/rc.d/",
    path => "/usr/bin:/usr/sbin:/bin"
            }
```

其他资源的用法请参考 puppet 官方文档，URL：

http://docs.puppetlabs.com/references/stable/type.html

第3章 Func 安装使用

3.1. Func 介绍

Func 全称为 Fedora Unified Network Controller（Fedora 统一网络控制器），是由 Fedara 社区维护的一款用于服务器自动化远程管理的工具。有如下特性：

- Func 可以在主控机上一次管理任意多台服务器，或任意多个服务器组；
- Func 基于 Certmaster（https://fedorahosted.org/certmaster/）建立了 Master – Slaves 主从 SSL 证书管控体系，可以将证书自动分发到所有受控服务器；
- Func 命令行可以直接发送远程命令或者远程获取数据；
- Func 开发者已经完成了大多数常用任务模块的开发：CommandModule、FileTrackerModule、JBossModule、IPtablesModule、HardwareModule、MountModule、NagiosCheck、NetappModule、NetworkTest、ProcessModule、ServiceModule、SysctlModule、RebootModule、RpmModule、VirtModule、YumModule 等等，这些模块的作用都可以顾名思义，或者参考：https://fedorahosted.org/func/wiki/ModulesList ；
- 任何人都可以通过 Func 提供的 Python API 轻松编写自己的模块，以实现具体功能扩展。而且
- 任何 Func 命令行能完成的工作，都能通过 API 编程实现；
- Func 通讯基于 XMLRPC 和 SSL 标准协议。

3.2. Func 安装与配置

Func 的 Master 端和 Slave 端安装方式相同，在安装过 EPEL 的 Centos 服务器中，可使用 yum –y install func 直接完成安装。

Master 的控制端与被控制端是由配置文件指定的，Master 配置，可以保持默认，如需打开证书自动分发功能，将 autosign 设为 yes 即可。

```
[root@server ~]# cat /etc/certmaster/certmaster.conf
# configuration for certmasterd and certmaster-ca
```

```
[main]
autosign = no
listen_addr =
listen_port = 51235
cadir = /etc/pki/certmaster/ca
cert_dir = /etc/pki/certmaster
certroot = /var/lib/certmaster/certmaster/certs
csrroot = /var/lib/certmaster/certmaster/csrs
cert_extension = cert
sync_certs = False
```

服务端配置完成后需要重启 certmaster 服务

被控制端需要手动配置/etc/certmaster/minion.conf 文件中的 certmaster 字段，用于指定控制端服务器的地址，如下所示：

```
[root@client1 ~]# cat /etc/certmaster/minion.conf
# configuration for minions

[main]
certmaster = puppet.puppet.cn #设置为 master 的 IP 或域名
certmaster_port = 51235
log_level = DEBUG
cert_dir = /etc/pki/certmaster
```

被控制端配置完毕后，使用 service funcd start 命令启动 funcd 服务后，就会自动向控制端提交证书申请请求。

在控制机中使用 certmaster-ca －l 查看到被控端的证书请求，用 certmaster-ca －s 主机名可以为相应的主机颁发证书，证书分发后，需要将 client 端的主机名加到 server 端的 hosts 文件中，如下表所示：

```
[root@jvpuppet certmaster]# certmaster-ca -l
No certificates to sign
```

```
[root@jvpuppet certmaster]# certmaster-ca -l
sxxb4-33
[root@jvpuppet certmaster]# certmaster-ca -s sxxb4-33
/var/lib/certmaster/certmaster/csrs/sxxb4-33.csr signed - cert located at
/var/lib/certmaster/certmaster/certs/sxxb4-33.cert
```

3.3. Func 使用说明

3.3.1.Func 服务器群组管理

Func 可以远程管理任意台或任意组服务器，对一台发送指令时，可直接指定其主机台，对所有服务器发送指令使用时，用*表示所有服务器，如下表所示：

```
1. 确认主机 web1 是否存在
[root@jvpuppet bin]# func "web1" ping
2. 让所有主机执行 ifconfig 命令
Func “*” call  command run “ifconfig”
```

日常的运维工作更多的对特定的服务器组进行操作，Func 提供了 GroupsApi 来对服务器进行分组管理，如下表所示：

```
添加一个新组 newgr :
[root@fedorabig func]# func "*" group --ag "newgr"
[root@fedorabig func]#
列出所有组及其服务器成员
[root@fedorabig func]# func "*" group --la
Group : group5
        Host : 1
        Host : 3
        Host : 2
        Host : 5
        Host : 4
Group : newgr
```

但用 groupapi 管理服务器组时比较繁琐，容易出错，Func 也支持用配置文

件/etc/func/groups 管理服务器群组及其成员，如下所示：

```
[webservers]
host = http1.example.com; http2.example.com
[mailservers]
host = mail1.example.com; mail2.example.com
```

3.3.2.Func 模块说明

Func 的开发者已经提供了常用的功能模块，用户也可以自己开发相应的功能模块，Func 官网目前提供的模块列表如下所示：

Modules List

We have a lot of core plugins so the command line and the API offer up some interesting features by default. As time goes on, the features ones will be added. Join the email list and submit your own!

- BridgeModule -- Allows for simple network bridge management
- CertMasterModule -- For power users out there, allows manipulating the certmaster from Func's API.
- CommandModule -- Running Arbitrary Commands Like SSH Does
- CopyFileModule -- Copyfile File Copying and Checksumming
- CpuModule -- Poll CPU statistics, varying time period allowed.
- DiskModule -- Get Disk information, currently just df output.
- FileTrackerModule -- tracks file changes, for use with FuncInventory
- GetFileModule -- Allow retrieving arbitrary files from minions
- JBossModule -- monitoring and control jboss instances
- IPtablesModule -- iptables management
- HardwareModule -- Hardware Profilling
- MountModule -- mount, unmount, and query mounted resources
- NagiosServerModule -- Lets you do things like schedule downtime or enable/disable alerts on hosts, hostgroups, and servicegroups
- NagiosCheck -- be able to call Nagios plugins and get their results, without needing to install nagios. Works with any plugin
- NetappModule -- Administer Netapp filers
- NetworkTest -- Test out network stuff.
- ProcessModule -- Process Info, memory usage, and Killing
- PullfileModule -- Pull a (list of) remote file(s) and save it locally
- ServiceModule -- Service Status and Control
- SysctlModule -- Configure kernel parameters at runtime
- RebootModule -- Reboot your system
- RpmModule -- for any distro that supports RPM, lists installed packages
- SmartModule -- Disk Smart (Hard Drive) Status
- UserModule -- we still need to implement this :)
- VirtModule -- works with koan, KVM, Xen, etc
- VlanModule -- Simple VLAN management
- YumModule -- for any distro that is yum based, installs packages

3.4. 常用模块使用说明

1. CPU Module

   查看 CPU 状态信息，使用方法

```
func target.example.org call cpu usage
func target.example.org call cpu jiffies
```

2. CommandModule

   远程执行命令，使用参数：

```
func target.example.org call command exists /bin/foo
func target.example.org call command run "/bin/foo arg1 arg2 arg3"
```

3. ServiceModule

服务管理，可以启动、停止服务和查看服务的状态，使用参数：

```
func target.example.org call service start httpd
func target.example.org call service stop httpd
func target.example.org call service status httpd
```

4. ProcessModule

进程管理，可以查看、杀掉进程，也可以查看每个进程的内存使用情况：

```
func target.example.org call process info "aux"
func target.example.org call process mem
func "*" call process pkill thunderbird -9
func "*" call process kill firefox-bin SIGHUP
```

以上几个模块为日常运维过程中最常用到的，其他模块的功能及使用方法可以参考：https://fedorahosted.org/func/wiki/ModulesList

第4章 附录

4.1. 参录资料

1. Pro Puppet - James Turnbull, Jeffrey McCune
2. puppet docs，URL：http://docs.puppetlabs.com/
3. Comparison of open source configuration management software，URL：http://en.wikipedia.org/wiki/Comparison_of_open_source_configuration_management_software
4. Func wiki，URL：https://fedorahosted.org/func/wiki/
5. Linux 运维趋势第 0 期

4.2. 招贤纳士

目前公司处于创业期，我们是一个快速学习、成长的技术团队。如果您热爱互联网，喜欢研究和探索新事物，享受在高压下不断成长的过程，喜欢和优秀的人一起工作，请联系我：

ID：netxfly

QQ：56219693

Mail：netxfly@hotmail.com