

# 自动化运维管理

自动化运维体系及开源产品对比  
大规模服务器自动化配置和管理  
Puppet开发与Func应用实例

XX中心 Hartnett

# 内容摘要

why

- 自动化运维的意义
- 自动化运维的背景

what

- 自动化运维体系介绍
- 常见开源产品介绍选型

how

- **Puppet**的安装、配置和使用
- **Func**的安装、配置和使用

e.g

- **Puppet**开发实例
- 自动化运维实例

# 自动化运维系统的意义与背景

## 为什么要使用自动化运维系统

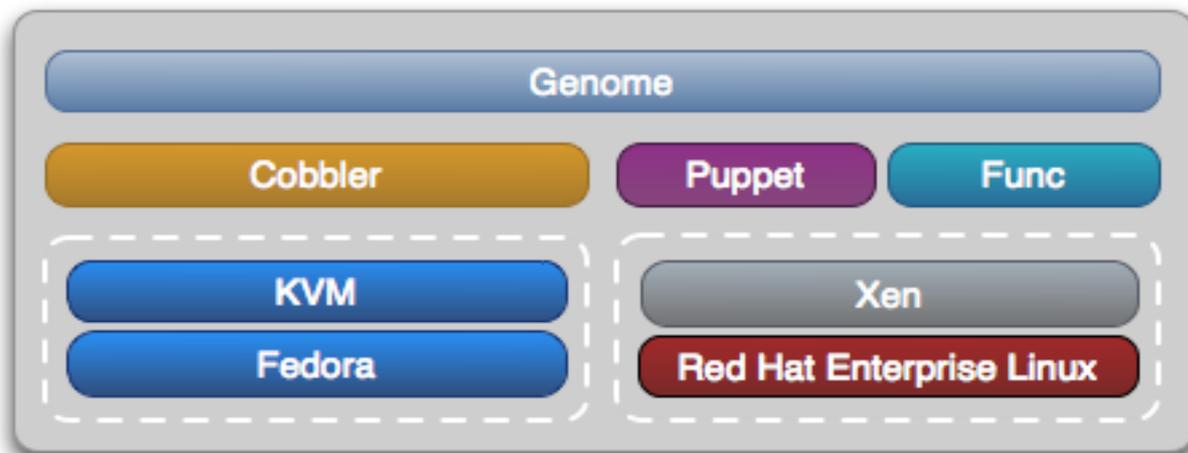


# 自动化运维体系

自动化运维体系结构	<h2>系统预备</h2>
	自动化安装操作系统及常用软件包
	<h2>配置管理</h2>
	<ol style="list-style-type: none"><li>1. 自动化部署业务系统软件包并完成配置</li><li>2. 远程管理服务器（开关服务等）</li><li>3. 变更回滚</li></ol>
	<h2>监控报警</h2>
	<ol style="list-style-type: none"><li>1. 服务器可用性、性能、安全监控</li><li>2. 向管理员发送报警信息</li></ol>

# 自动化运维工具介绍

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



# 自动化运维工具对比选型（1）

## 预备类工具

### Kickstart

1. 供Linux操作系统安装管理器Anaconda读取的无人值守安装配置脚本
2. 安装配置过程较为繁琐

### Cobbler

1. 一个集成工具，集成了PXE、DHCP、DNS和Kickstart服务管理和yum仓库管理
2. 简化了运维工程师工作量

# 自动化运维工具对比选型（2）

## 配置管理类工具

### Chef

1. 学习门槛高（ruby）
2. 脚本维护调试繁琐
3. 依赖包多，配置过程复杂繁琐
4. chef的配置管理文件二进制文件中，维护不方便
5. chef的用户群少，出了问题不方便排查

### Puppet

1. 入门简单，管理模块开发周期短（puppet语言、资源）
2. 脚本维护调试方便
3. 安装、配置简单
4. 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

# 自动化运维工具对比选型（3）

## 配置管理类工具

### Cfengine

1. 老牌的配置管理工具，功能强大
2. 语法晦涩难懂，学习、维护成本高

### Puppet + Func

1. 新兴的配置管理工具，语法简单，易于学习、维护
2. 远程执行命令只能返回成功与否，执行过程无法跟踪查看
3. Linux集群管理工具Func可以弥补Puppet远程执行命令的不足

# 自动化运维工具对比选型（4）

## 监控报警类工具

### Zabbix

1. Zabbix、Nagios和cacti等工具，zabbix和Nagios+cacti组合都是很优秀的工具
2. 鉴于zabbix参考资料较少，选择了常用的Nagios+cacti组合

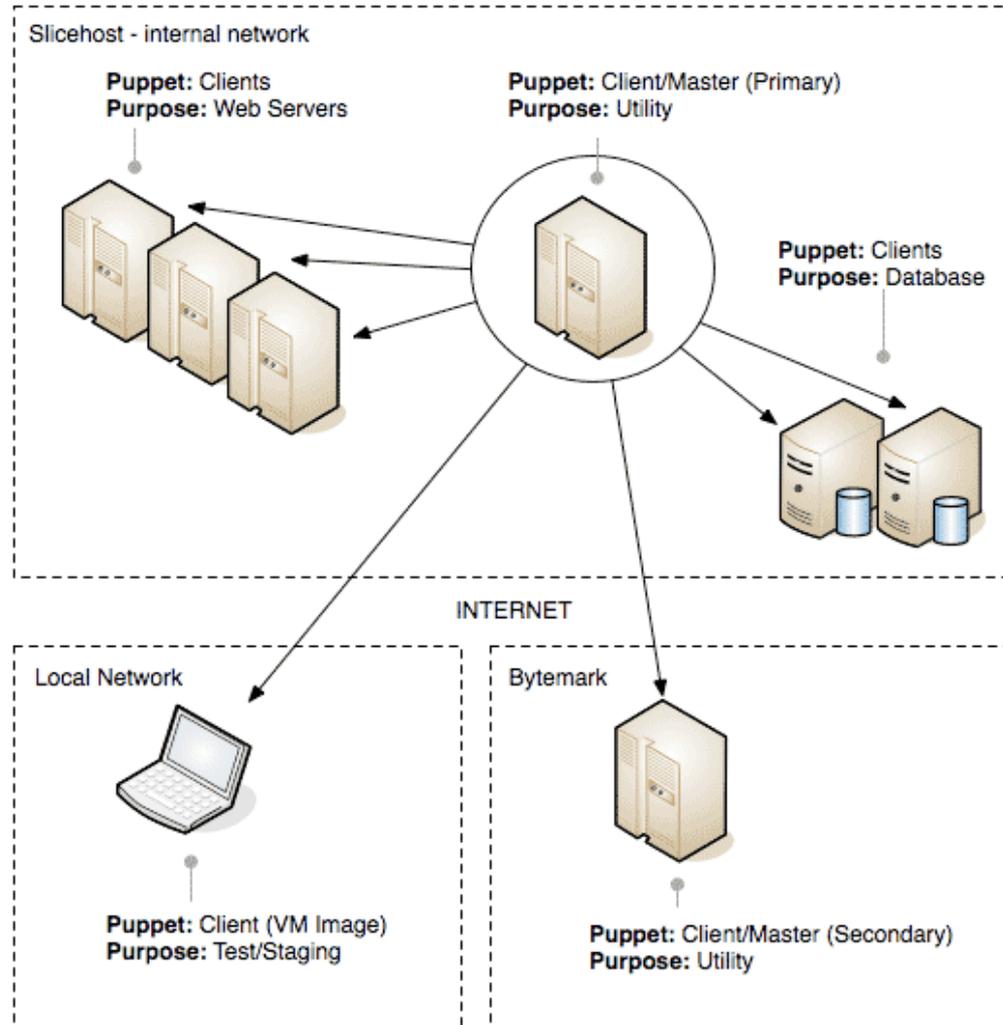
### Nagios + Cacti

1. Nagios擅长服务器可用性监控，报警功能强大
2. Cacti擅长监控历史数据收集、存储
3. UI美观，数据展示功能强大

# Puppet篇章—Puppet介绍

- Puppet Labs基于ruby语言开发
- 支持以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
- 支持对Windows操作系统管理（功能有限）
- 适用于业务系统的整个生命周期
- Clients默认每30分钟请求一次Server端
- 支持以节点的方式管理若干的服务器群组

# Puppet篇章—Puppet架构



# Puppet篇章—Puppet Server安装

- 安装说明

- 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（注意32位与64位版本区别）

- 安装方法

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`

# Puppet篇章—Puppet Client安装

Puppet服务器端对客户端的管理是基于主机名的，在安装之前需要为客户端设置唯一的主机名

## ● 安装过程

1. 安装EPEL包
2. 安装puppet客户端：  
`yum -y install puppet`
3. 提交证书申请  
`puppetd --server jvpuppet.jooov.cn -test`

# Puppet篇章—Puppet 证书管理

- 服务器端可用puppetca管理证书
  1. puppetca -list , 查看到申请证书的客户端主机名
  2. puppetca -s 主机名 , 为特定的主机颁发证书
  3. puppetca -s and -a , 为所有的主机颁发证书
  4. puppetca -l and -a , 列出所有主机 , +号代表已经领证的机器

```
[root@jvpuppet ~]# puppetca -l and -a
+ hartnett-test1
+ jvpuppet.jooov.cn
+ netbanktest.localdomain
+ sjn45-140.localdomain
+ sjn45-142.localdomain
+ sjn45-145.localdomain
```

# Puppet篇章—Puppet 使用

- 默认安装好的Puppet没有任何配置管理功能，需要运维人员自行开发配置管理模块

# Puppet开发—Server端目录结构

- Server端默认安装后，位于/etc/puppet目录下

---

```
[jvuser@jvpuppet puppet]$ pwd
```

```
/etc/puppet
```

```
[jvuser@jvpuppet 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`入口脚本文件

`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

---

# Puppet开发—Puppet脚本开发规范

```
[jvuser@jvpuppet puppet]$ tree
|-- auth.conf
|-- fileserver.conf
|-- manifests ← puppet脚本主目录
    |-- modules.pp
    |-- nodes
        |-- bjq.pp
        |-- bjzw.pp
        |-- sjn.pp
        |-- smzqc.pp
        |-- sxxbd-05-09.pp
        `-- test.pp
    `-- site.pp ← 入口模块文件
|-- modules ← 功能模块目录
    |-- func
        |-- README
        |-- manifests
            |-- base.pp
            `-- init.pp
        |-- templates
        `-- minion.conf
    |-- installio
        |-- manifests
        `-- init.pp
    |-- templates
```

# Puppet开发—Puppet资源介绍

- Puppet提供了48种资源类型，支持用户自主开发

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）
<u>Crontab</u> 任务	<u>Crontab</u> 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）
<u>Zfs</u>	管理 ZFS，在 <u>zfs</u> 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

# Puppet资源范例—file资源

- **功能：**  
新建一个/tmp/puppettest文件，文件内容为：puppet test only，文件权限为666
- **代码：**

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

# Puppet资源范例—用户和组资源

- 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组

## 代码：

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

# Puppet资源范例—Package资源

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

- **代码：**

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

# Puppet资源范例—Service资源

- **功能：**  
以下实例为启动ssh服务，停止nfs服务

- **代码：**

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

# Puppet资源范例—crontab资源

- **功能：**  
在crontab添加时间同步任务

- **代码：**

cron

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

# Puppet资源范例—exec资源

- **功能：**

自动执行初始化iptables安全策略脚本

- **代码：**

```
exec {  
    "/etc/rc.d/jv_firewall.sh":  
    cwd => "/etc/rc.d/",  
    path => "/usr/bin:/usr/sbin:/bin"  
}
```

# Func篇章—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
- **特点：**
  1. 可以管理任意多台服务器，或任意多个服务器组
  2. 基于 Certmaster建立主从 SSL 证书管控体系
  3. 提供了多个实用的远程管理模块
    - CommandModule
    - ProcessModule
    - ServiceModule
  4. 任何 Func 命令行能完成的工作，都能通过python API 编程实现

# Func篇章—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
- 完成配置后需要重启certmaster服务

# Func篇章—Func Server端配置

# 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

# Func篇章—Func Client端配置

- Client端需要配置/etc/certmaster/minion.conf中的certmaster字段，用于指定Server的证书地址

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

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

# Func篇章—Func 证书管理

- **证书申请：**  
Client配置完毕后，使用service funcd start启动funcd服务后，会自动向控制端提交证书申请请求
- **证书颁发：**
  1. 在Server端执行certmaster-ca -l查看到被控端的证书请求
  2. 用certmaster-ca -s 主机名可以为相应的主机颁发证书
  3. 证书分发完后，需要在server的hosts中加client的主机名
- **实例：**

```
[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
```

# Func使用—远程管理命令执行方法

- Func可以远程管理任意台或任意组服务器，func同时支持命令行和python api调用
- 对一台发送指令时，可直接指定其主机名  
**确认主机jooov-web是否存在：**
  1. `[root@puppet bin]# func "jooov-web" ping`
  2. `client = func.Client("jooov-web")`  
`print client.ping()`
- 对所有服务器发送指令时，用\*表示所有服务器  
**让所有主机执行ifconfig命令：**
  1. `func "*" call command run "ifconfig"`
  2. `client = func.Client("*")`  
`print client.command.run("ifconfig")`

# Func使用—服务器群组管理

- 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2种方法对服务器进行分组管理

## 1. GroupApi

添加一个新组newgr :

```
[root@fedorabig func]# func "*" group --ag "newgr"
```

列出所有组及其服务器成员

```
[root@fedorabig func]# func "*" group -la
```

## 2. 组管理配置文件 ( /etc/func/groups )

```
[webservers]
```

```
host = http1.example.com; http2.example.com
```

```
[mailservers]
```

```
host = mail1.example.com; mail2.example.com
```

# Func使用—功能模块说明

- Func服务器管理功能是由官方提供的功能模块实现的

## 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 Profiling
- [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

# Func使用—常用功能模块使用说明

- CPU Module  
查看CPU状态信息，使用方法  
`func target.example.org call cpu usage`
- CommandModule  
远程执行命令，使用方法：  
`func target.example.org call command run "ifconfig -a"`
- 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 status httpd`
- ProcessModule  
进程管理，可以查、杀进程，也可以查看每个进程的内存使用情况：  
`func target.example.org call process info "aux"`

# 内容回顾 && 提问时间

